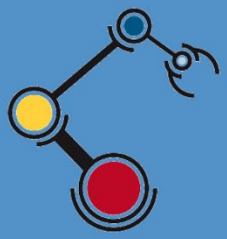
Escuela Politécnica Superior

Teleoperación de un brazo robot Kinova MICO2 a través de un dispositivo Omni Bundle



Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

Autor:
Alex Darwin Paredes Anchatipán

Tutores:
Andrés Úbeda Castellanos
Gabriel Jesús García Gómez

Universitat d'Alacant
Universidad de Alicante

Junio 2018

UNIVERSIDAD DE ALICANTE

TRABAJO FIN DE MÁSTER

Teleoperación de un brazo robot Kinova MICO2 a través de un dispositivo Omni Bundle

Autor
Alex Darwin Paredes Anchatipán

Tutores
Andrés Úbeda Castellanos
Gabriel Jesús García Gómez

Trabajo presentado previo a la obtención del título de

Máster en Automática y Robótica
Departamento de Física, Ingeniería de Sistemas, y Teoría de la Señal

4 de junio de 2018

Resumen

El avance tecnológico en materia de telecomunicaciones ha logrado romper la brecha de la distancia alrededor del mundo, permitiéndonos compartir información en tiempos muy cortos y haciendo posible realizar tareas como mensajería instantánea, videoconferencias, monitoreo remoto, etc. En el ámbito de la robótica ha abierto varios campos de investigación, tales como la teleoperación y la telepresencia, con aplicaciones que van desde un uso doméstico hasta fines médicos o incluso espaciales.

En este trabajo se plantea efectuar el control remoto para entornos cotidianos de un brazo robótico *Kinova MICO2* comandado por un dispositivo háptico *Phantom Omni* de la empresa *3D Systems*, con realimentación de fuerza, dotando así de sensibilidad al operador. Se estudiará el método de control más efectivo de acuerdo a los requerimientos del sistema, teniendo opciones de control en espacio cartesiano y articular y éstos a su vez por posición o velocidad. Finalmente se implementará una interfaz de usuario mediante *Visual Studio* para el monitoreo y la configuración de los instrumentos utilizados.

Se emplearán las herramientas ofrecidas por los fabricantes, siendo éstos *KINOVA SDK* y *OpenHaptics* para el brazo robótico y el mecanismo háptico respectivamente. Como protocolo de comunicación, se establecerá sobre UDP, siendo la mejor opción en cuanto a velocidad y alcance.

Agradecimientos

Agradezco a las personas que han sido los pilares a lo largo de mi vida, mi familia. A mis tutores Andrés y Gabriel que han mostrado toda la predisposición y ayuda necesaria para finalizar este proyecto, a mis compañeros del máster por la amistad forjada durante el curso, a los profesores y a la Universidad de Alicante en general por la acogida que me han brindado.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo se describirá como está estructurado el documento, una exposición global de la temática a investigar junto con el estado actual del mismo, y finalmente se planteará la propuesta y objetivos a cumplir en este proyecto.

1.1. Estructura

La estructura que se va a seguir en este manuscrito corresponderá de manera general al orden en el que se desarrolló el trabajo práctico, en consecuencia, se abordará en el primer capítulo la teoría y los trabajos actuales dentro del campo de investigación, y los propósitos a cumplir.

En el capítulo 2 se presentará el procedimiento llevado a cabo durante todo el desarrollo experimental, primero como dispositivos independientes, seguido de la integración de ambos, el protocolo de comunicación para la teleoperación y la interfaz de usuario final, continuando con el tercer capítulo en el cual se expondrá los experimentos llevados a cabo, tanto en tipo de control como parámetros utilizados hasta llegar al resultado deseado.

Finalmente se plasmará las conclusiones a las que se ha llegado con este proyecto y se propondrá trabajos futuros que puedan derivar de esta investigación.

1.2. Descripción general

Al pensar en la palabra robot inmediatamente surgen dos distinciones importantes: los de tipo industrial y los destinados al entorno doméstico o comercial. En este proyecto se trabajará con el robot MICO2 fabricado por la empresa Kinova, enfocado en la asistencia a personas que requieran ayuda para interactuar con el entorno cotidiano, siendo éstos, brazos colaborativos con aspecto amigable y dotados de muñecas con dedos actuados para cumplir con tareas diarias de un ser humano, y generalmente controlados por un *joystick*, lo cual suple la mayoría de aplicaciones; pero esto implica que el operario debe tener la capacidad de realizar tales acciones de mando y encontrarse cerca del mismo, en el caso de que

la persona no sea capaz o no tenga las condiciones necesarias para hacerlo, o a su vez, por la naturaleza de la tarea o por seguridad, no deba encontrarse cerca del área de trabajo del robot, este tipo de control no sería el adecuado.

Para atender a este problema, surge la teleoperación como solución, con lo cual se puede brindar asistencia remota a personas que por cualquier motivo no puedan operar un brazo robótico por sus propios medios, o para aplicaciones que estén obligadas a mantener distancia entre el operario y el artefacto robótico.

Una vez hallada la solución, surge el cómo desarrollar la misma, a priori lo más lógico sería recrear el *joystick* ya sea física o virtualmente y mediante alguna técnica de telecomunicación enviar los comandos respectivos para el movimiento, pero este método es poco intuitivo para quien lo opera y de menor eficiencia en cuanto a posicionar el robot en el espacio. Por tanto, se propone utilizar un dispositivo háptico como mando de control, resultando más cómodo y natural de manejar. Además, aprovechando los sensores del robot MICO2 y la característica de los dispositivos hápticos de generar perceptibilidad, se puede aumentar la telepresencia mediante realimentación de fuerzas, dotando así de sensibilidad al ejecutor sobre el brazo actuado.

Finalmente, para otorgar un entorno más amigable al consumidor final, se plantea realizar una interfaz de usuario para el sistema operativo *Windows*, que se encargará del monitoreo de ambos dispositivos, así como de ciertas configuraciones para su correcto funcionamiento.

1.3. Estado del arte

En este apartado se estudiará los conceptos, técnicas y evolución que ha tenido la teleoperación y la telepresencia, así como trabajos relacionados con el objetivo de esta investigación.

1.3.1. Definiciones

Antes de desarrollar el tema, es pertinente aclarar algunos significados básicos, que serán usados posteriormente.

Teleoperación: conjunto de tecnologías que comprenden la operación a distancia de un dispositivo por un ser humano. En consecuencia, teleoperar es la acción que realiza una persona para operar a distancia un aparato, denominado dispositivo teleoperado [1].

Telepresencia: es el uso de sistemas para trasladar virtualmente al usuario a otra locación, es decir, representa la capacidad de interacción con un entorno real y remoto desde la perspectiva del usuario [2].

Telerrobótica: conjunto de tecnologías que comprenden el monitoreo y reprogramación de un robot por un ser humano. Por tanto, se hablará de telerrobot a la teleoperación de un artefacto robótico [1].

Realimentación táctil: retroalimentación de la sensación de contacto aplicada a la piel. Permiten conocer la geometría de la superficie, su rugosidad y su temperatura [1].

Realimentación cinestésica o de fuerzas: retroalimentación de la sensación de resistencia al avance o a un peso. Proporcionan información sobre la fuerza de contacto, así como el peso y deformabilidad de un objeto [1].

Realimentación háptica: retroalimentación de la sensación de contacto, ya sea táctil o de fuerzas [1].

Robot de asistencia: mecanismo robótico que brinda ayuda o soporte a un usuario humano. Siendo algunos ejemplos: robots para rehabilitación, de compañía, brazos manipuladores, etc. [3].

1.3.2. Antecedentes

Como la mayoría de innovaciones tecnológicas, la telemanipulación surge de una necesidad, en este caso, la manipulación de objetos peligrosos en la industria nuclear. Dando lugar al primer manipulador teleoperado mecánico, en 1948, denominado M1. Este mecanismo consistía en reproducir fielmente los movimientos realizados por el operario en el manipulador maestro sobre la pinza ubicada en el manipulador esclavo. Cabe destacar que ambos aparatos eran prácticamente iguales, por tanto, el movimiento podía ser imitado eje a eje, dando lugar a una trayectoria igual entre ambos [1]. Este tipo de teleoperación se muestra en la Figura 1.1.

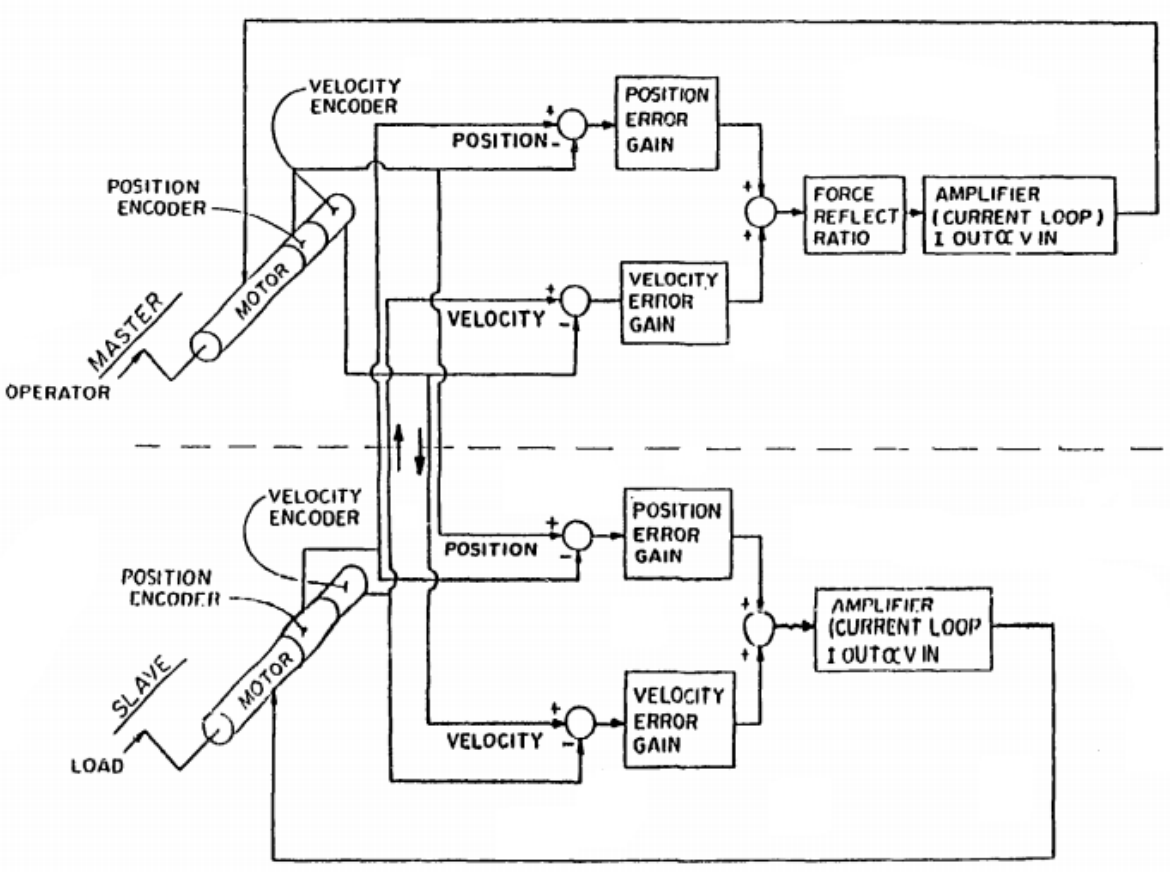


Figura 1.1. Diagrama de bloques de un servomecanismo bilateral. Fuente: [4].

Desde el planteamiento de este tipo de sistemas, se ha ido desarrollando más la tecnología, métodos y aplicaciones. Actualmente no es necesario que el dispositivo maestro y esclavo sean iguales, pudiendo ser uno la escala del otro, o incluso con modelos cinemáticos completamente diferentes, como es el caso de este estudio.

Un sistema de teleoperación común, consta de las siguientes partes [2]:

- Un dispositivo maestro (interfaz háptica), la que sostiene y manipula la persona.
- Un dispositivo esclavo, que es el encargado de la manipulación de acuerdo a los comandos enviados por el maestro.
- Un controlador, que acopla ambos el dispositivo maestro y esclavo transmitiendo movimientos y fuerzas entre los mismos.

Un sistema de control de teleoperación, tiene como objetivos [1]:

- Hacer que el control manual del operador sea robusto ante retardos, saturación de los actuadores y otras no linealidades e incluso a errores propios del ser humano.

- Elevadas prestaciones en la teleoperación, haciendo que los bucles de control tengan un comportamiento dinámico apropiado y reduciendo el trabajo del operador.

1.3.3. Control bilateral

Para entender el concepto de control bilateral, primero hay que definir al control unilateral, siendo este cuando el dispositivo maestro genera las señales de referencia, ya sea posición o velocidad, para el lazo de control del dispositivo esclavo [1], lo que significa que las perturbaciones del entorno no tienen respuesta en la persona que está realizando el control.

Por consiguiente, el control bilateral además de abarcar las funciones del método anterior, agrega realimentación de fuerzas hacia el operario, de modo que se da un lazo totalmente cerrado entre ambos mecanismos, ya que si bien el dispositivo maestro es quien gobierna las acciones del dispositivo esclavo, éste último también es capaz de ejercer acciones sobre el instrumento maestro y a su vez, a la persona operante. El concepto se ilustra en la Figura 1.2.

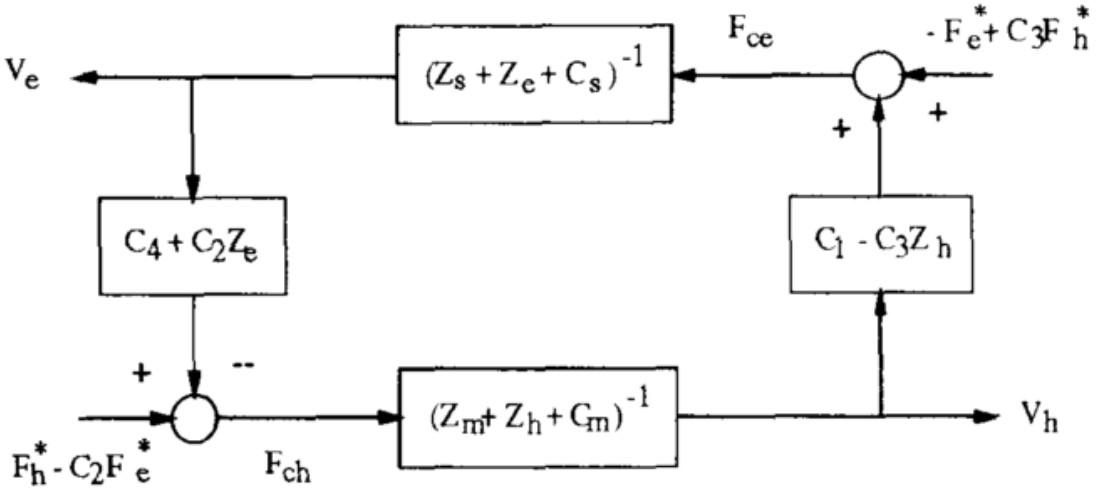


Figura 1.2. Representación de un lazo de control bilateral. Fuente: [5].

Al ser un sistema teleoperado, implica una separación física entre el manipulador y el actuador, haciendo que necesariamente aparezca un tiempo de retardo entre las acciones y la respuesta entre ambos dispositivos, que dependerá principalmente de la comunicación, lo que habrá que tener en cuenta para diseñar e implementar los respectivos lazos de control, ya que retardos excesivos o pérdidas de comunicación podrían causar un comportamiento inestable en el mecanismo esclavo.

En la Figura 1.3 se exhibe un diagrama de bloques que contiene todas las etapas a considerar para un sistema de teleoperación con realimentación de fuerzas, empezando por el operador humano que es quien gobierna las acciones, seguido del mecanismo maestro, encargado de interpretar las mismas, a continuación está el retardo debido a la comunicación. En el lado del esclavo tenemos el controlador de posición que actuará sobre el dispositivo teleoperado, y éste a su vez sobre el entorno que al interactuar con el manipulador generará respuestas de fuerza hacia el operante, siguiendo el camino inverso antes descrito.

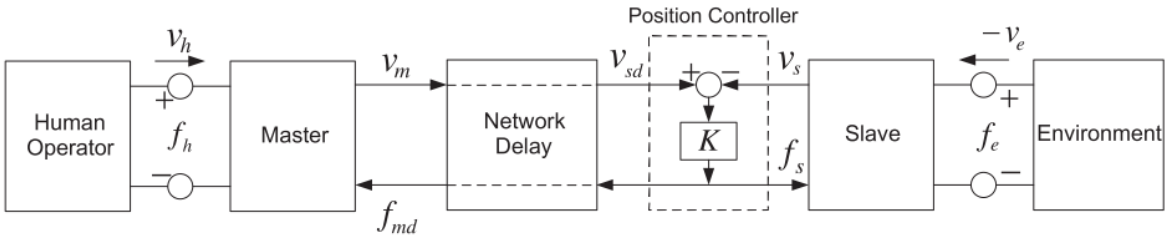


Figura 1.3. Diagrama de bloques de un sistema de teleoperación bilateral con tiempo de retardo. Fuente: [6].

1.3.4. Dispositivo háptico

Puede extraerse una definición de [2], como un dispositivo compuesto de partes mecánicas, que trabaja en contacto físico con el cuerpo humano con el propósito de intercambiar información. Cuando se trabaja con un dispositivo háptico el usuario transmite comandos mediante la manipulación física, y éste a su vez, crea una sensación háptica al usuario mediante la correcta estimulación de los sistemas sensoriales.

Por tanto, no existe una configuración única para un equipo háptico. En la Figura 1.4 se presenta una clasificación basada en el espacio de trabajo, fuerza y precisión. Existen dispositivos pequeños, aptos para la muñeca o la mano, de gran precisión, pasando a mecanismos que intentan recrear todo el brazo de un ser humano, e incluso móviles de tipo exoesqueleto, y por el otro lado, mecanismos para aplicaciones industriales, que requieren mayor fuerza y potencia.

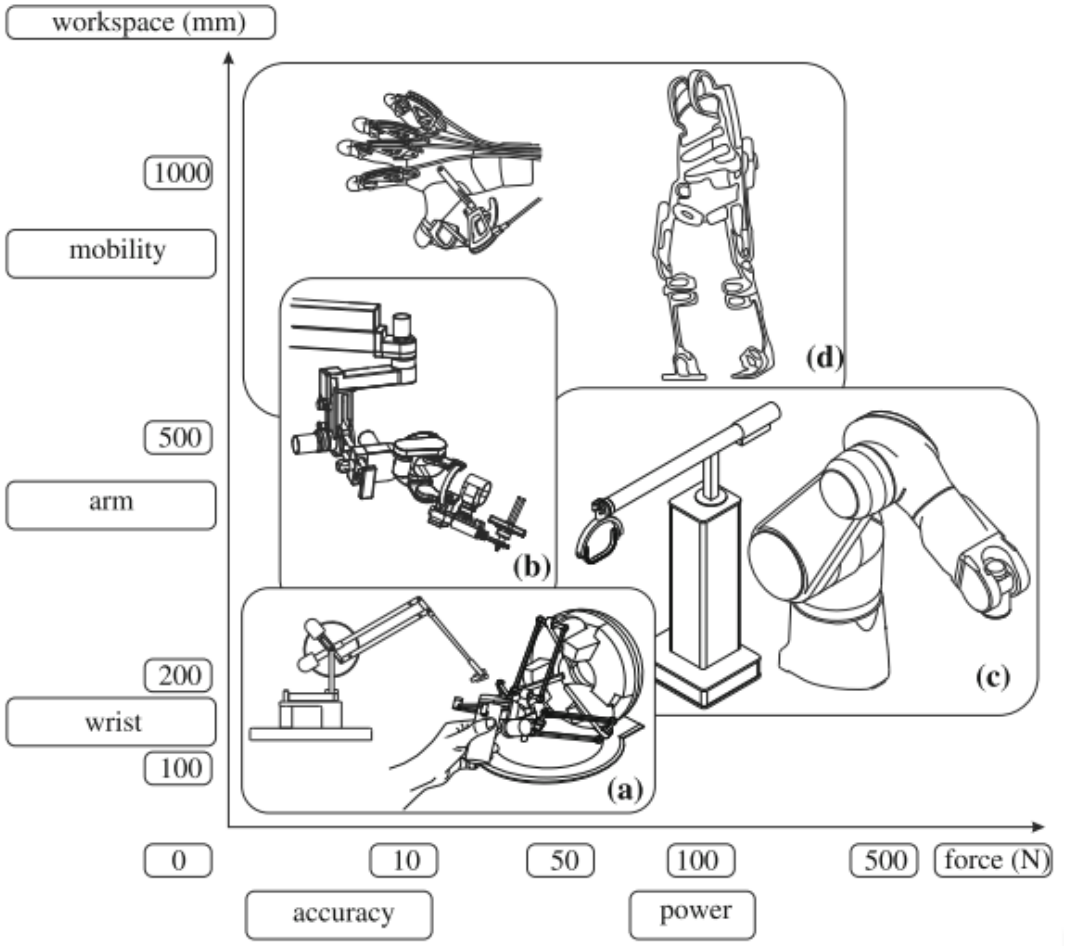


Figura 1.4. Clasificación de dispositivos hápticos según su espacio de trabajo, potencia y precisión. (a) Muñeca y mano, (b) exoesqueleto de un brazo, (c) manipuladores industriales. (d) Móviles. Fuente: [2].

1.3.5. Brazo robótico

La RAE (Real Academia Española) define el término robot como: "Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas". Mientras que el RIA (*Robotics Institute of America*) lo define como: "Un manipulador reprogramable y multifuncional diseñado para mover materiales, partes, herramientas o dispositivos especiales, por medio de varios movimientos programados para la realización de diversas tareas o trabajos".

De los conceptos antes mencionados, podemos describir un brazo robótico como un mecanismo electrónico con forma antropomórfica análoga a un brazo humano diseñado para manipular objetos y realizar diversas tareas o trabajos.

Si bien el diseño puede variar y ser único en cada brazo robótico, un esquema general y ampliamente usado se muestra en la Figura 1.5, donde se expone un

manipulador de 6 grados de libertad, separado en dos secciones: el brazo, que nos otorga posicionamiento en el espacio, y la muñeca, que nos brinda orientación respecto al punto deseado.

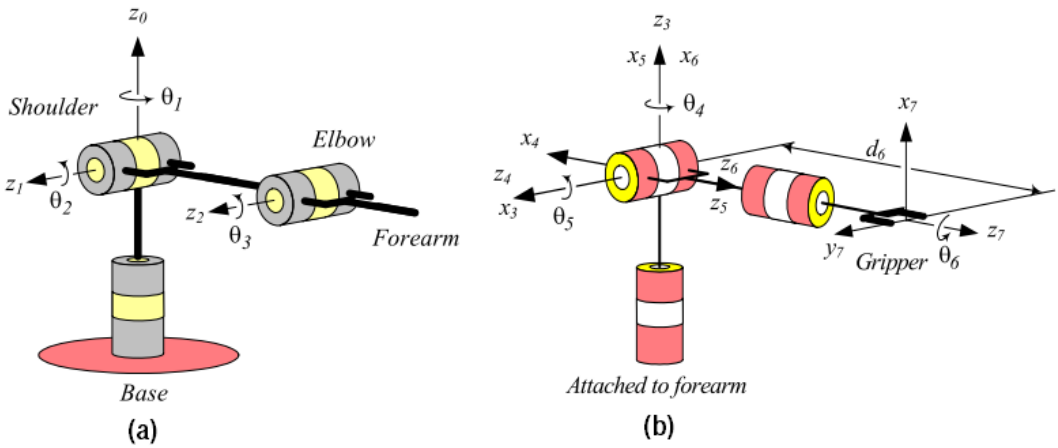


Figura 1.5. Esquema general de un manipulador con 6 articulaciones rotacionales. (a) Brazo, (b) Muñeca. Fuente: [7].

1.3.6. Aplicaciones y trabajos relacionados

El campo de aplicación para esta tecnología es muy variado, por lo cual, se va a presentar algunos ejemplos destacados.

Industria nuclear

Al ser la industria pionera, y por la naturaleza peligrosa de sus elementos, la teleoperación está muy extendida en este sector, desde manipulación de materiales delicados con robots estáticos [4], hasta exploración con robots móviles, como ejemplo, el robot *Pioneer*, mostrado en la Figura 1.6, diseñado para inspeccionar y evaluar el reactor nuclear de la planta de Chernobyl [8].

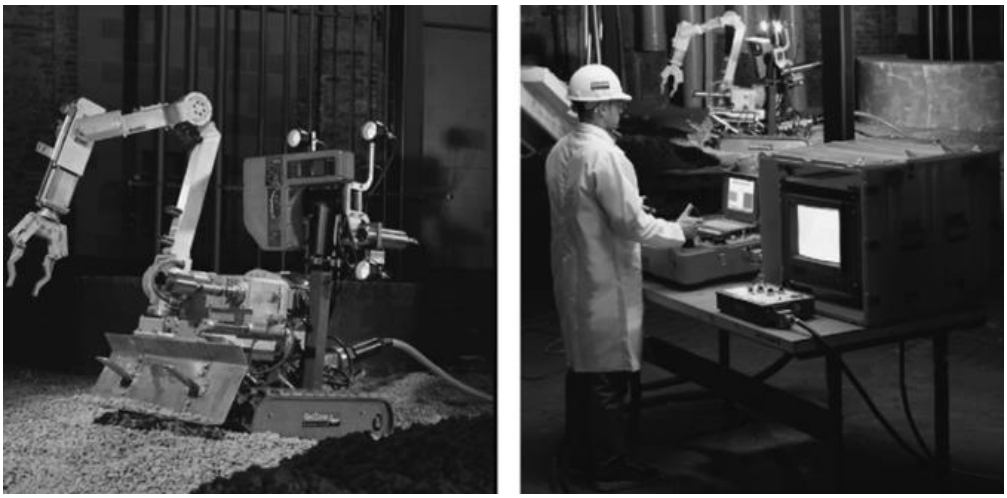


Figura 1.6. Robot *Pioneer* (izquierda), y estación de control (derecha). Fuente: [8].

Aplicaciones militares

Siendo la teleoperación una manipulación a distancia, es de esperar que haya encontrado su uso en el campo militar, permitiendo maniobrar robots y armas desde un lugar seguro. Pueden ser vehículos terrestres, aéreos o marinos. En la Figura 1.7 se exhibe la fotografía de un robot TAROS.

Figura 1.7. Robot TAROS (*Tactical Robotic System*). Fuente: [9].

Sector Industrial

Se puede mencionar el trabajo realizado por Ni, et al. [10], que consiste en utilizar realidad aumentada, conjuntamente con dispositivos hápticos *PHANToM* para programar robots de soldadura. En la Figura 1.8 se observa los resultados obtenidos de la investigación.

Figura 1.8. Visión VR e interfaz háptica en uso industrial. Fuente [10].

Aplicación espacial

La primera aplicación de teleoperación espacial se dio en la década de los setenta con el robot soviético *Lunokhod* mostrado en la Figura 1.9, para explorar el terreno lunar. A partir de ahí se ha seguido utilizando este tipo de robots en la exploración espacial, como la NASA en expediciones a Marte [8].

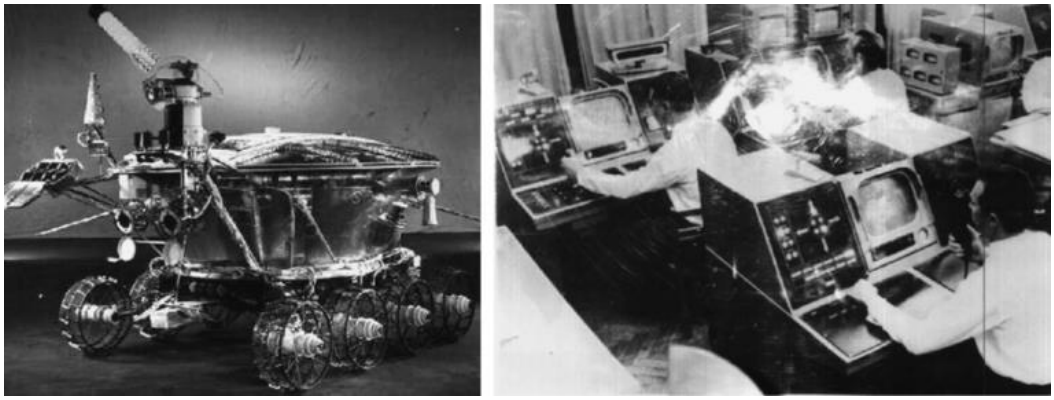


Figura 1.9. *Rover Lunokhod* a la izquierda, y la estación de control a la derecha. Fuente: [8].

Medicina

Probablemente sea uno de los campos que más atención ha recibido por parte de investigadores, se podría citar muchas aplicaciones en este campo, y por mencionar alguno ya que está relacionado con esta investigación por el dispositivo háptico utilizado y realimentación de fuerza, se hará alusión al trabajo de Olivieri et al [11], en el que se propone un sistema para operación láser como ilustra la Figura 1.10.

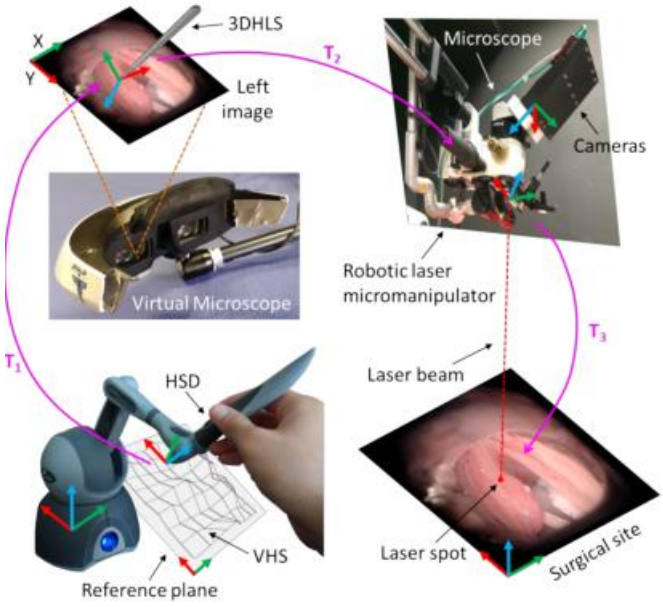


Figura 1.10. Sistema de cirugía láser con retroalimentación háptica. Fuente: [11].

1.4. Propuesta y objetivos

La propuesta de este proyecto es utilizar el instrumento háptico *Phantom Omni* como dispositivo maestro para teleoperar un brazo robótico *MICO2* en entornos cotidianos, con realimentación de esfuerzos para dotar de sensibilidad del entorno al operador. Esto involucra cumplir los siguientes objetivos:

- Integrar los dispositivos háptico y robótico mediante las herramientas de desarrollo ofrecidas por sus respectivos fabricantes.
- Diseñar e implementar una interfaz gráfica de usuario para el monitoreo y configuración del sistema de teleoperación.
- Efectuar los sistemas de control para el robot teleoperado, así como para las acciones del manipulador.
- Validar el sistema mediante experimentos que demuestren el correcto funcionamiento del mismo.

Capítulo 2

Metodología

En este capítulo se expondrá el procedimiento seguido para cumplir con los objetivos y la propuesta planteada para este proyecto. Esto contempla desde la investigación previa de los dispositivos a utilizar, fundamentos matemáticos y experimentación hasta llegar al resultado final.

2.1. Dispositivo háptico

El artefacto háptico utilizado para esta investigación es el modelo *Phantom Omni* desarrollado por *SensAble Technologies*, ahora llamado *Geomagic Touch* por la empresa *3D Systems,* quienes lo comercializan actualmente. Posee seis grados de libertad, de los cuales, los tres primeros otorgan la ubicación y conjuntamente son actuados para dar la sensación háptica de fuerzas y los tres siguientes utilizados para la orientación, sin actuadores. Además, cuenta con dos botones físicos, capaces de ofrecer tres funciones, con pulsaciones independientes y mediante la combinación de ambos. Se puede observar una fotografía obtenida de la guía de usuario en la Figura 2.1.

Figura 2.1. *Phantom Omni* o *Geomagic Touch*. Fuente: [12].

2.1.1. Características del dispositivo Phantom Omni

Lo primero a señalar es el diagrama cinemático que usa originalmente el *Phantom Omni*, y así poder entender de mejor manera sus especificaciones. En la Figura 2.2 se ilustra el marco de referencia, así como también sus grados de libertad.

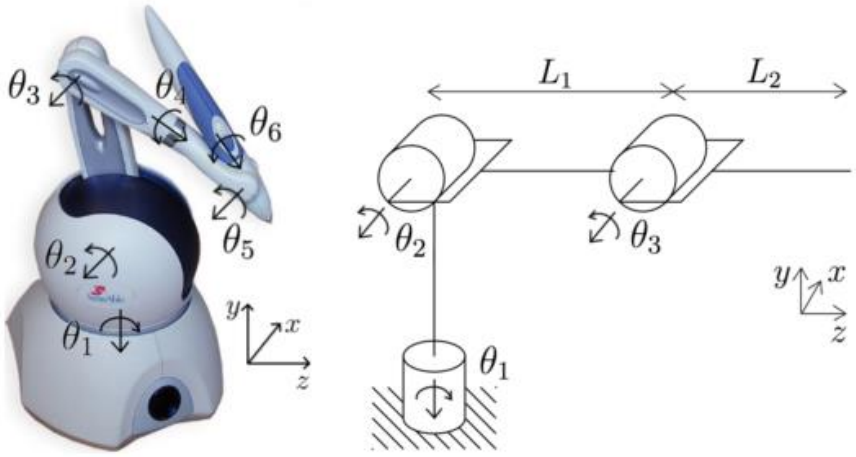


Figura 2.2. Diagrama cinemático para el *Phantom Omni*. Fuente: [13]

De las especificaciones mostradas en la Figura 2.3, las más destacables para este proyecto son: la fuerza que es capaz de ejercer, de 3.3 Newtons; la resolución en ubicación de aproximadamente 0.055 milímetros; y el espacio de trabajo, que si bien se indica numéricamente, para mayor comprensión se presenta un gráfico del mismo en la Figura 2.4.

SPECIFICATIONS	TOUCH™
Workspace	~6.4 W x 4.8 H x 2.8 D in > 160 W x 120 H x 70 D mm
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	> 450 dpi ~0.055 mm
Maximum exertable force and torque at nominal position (orthogonal arms)	0.75 lbf/3.3 N
Stiffness	x-axis > 7.3 lb/in (1.26 N/mm) y-axis > 13.4 lb/in (2.31 N/mm) z-axis > 5.9 lb/in (1.02 N/mm)
Force feedback (6 Degrees of Freedom)	x, y, z
Position sensing/input (6 Degrees of Freedom) [Stylus gimbal]	x, y, z (digital encoders) [Roll, pitch, yaw (± 5% linearity potentiometers)]
Interface	USB 2.0

Figura 2.3. Especificaciones del *Phantom Omni*. Fuente: [12].

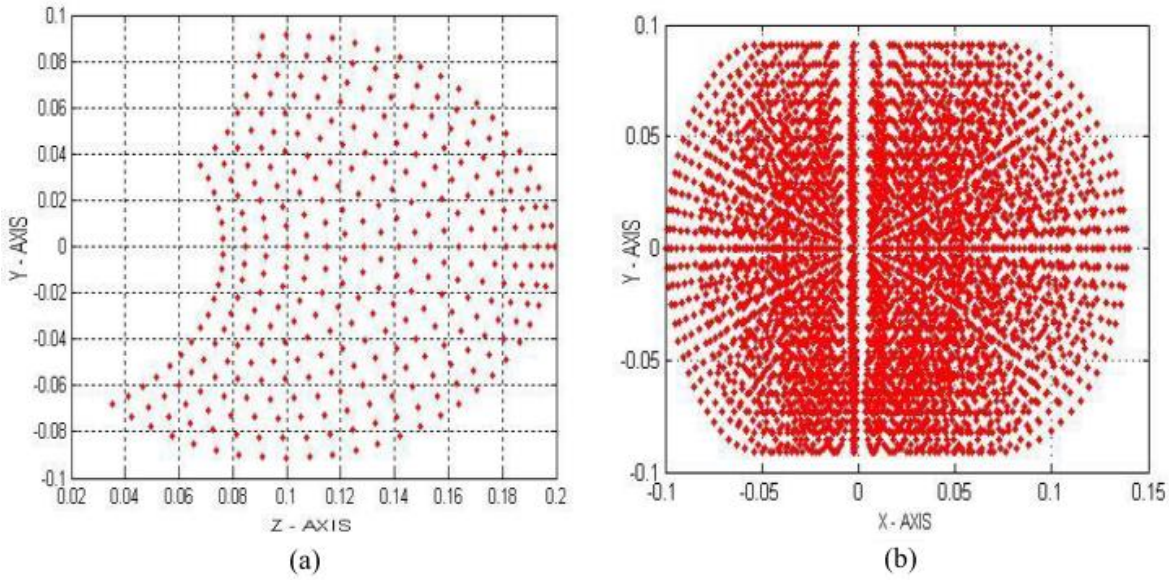


Figura 2.4. Espacio de trabajo del *Phantom Omni*. (a) Plano Y – Z, (b) Plano X – Y. Fuente: Modificado de [14].

2.1.2. Parametrización con el método de Denavit-Hartenberg

Si bien ya existen estudios en los que se detalla ecuaciones para describir el punto final del *phantom* [13], [15], se lo hace simplificando el modelo a tres grados de libertad o por el método geométrico, por tanto, en este trabajo se propone caracterizar todo el dispositivo mediante el método de Denavit-Hartenberg, lo cual es conveniente para simular y visualizar todo el mecanismo, y no únicamente su punto final o efector. En la Figura 2.5 se expone las consideraciones tomadas para la parametrización, cabe remarcar que se ha añadido una distancia *l1* desde el origen absoluto hasta el origen real del aparato, en caso de que sea necesario desplazar la altura de trabajo; además se ha cambiado el marco de referencia para coincidir con el sistema de referencia usado por el brazo MICO2. Igualmente se incluye las medidas utilizadas y el rango de giro de cada articulación en la Tabla 2.1.

Eslabón	Distancia [m.]	Articulación	Rango de movimiento [°]
l1	0	**q1**	~ -55 a 55
l2	0.15	**q2**	~ 0 a 100
l3	0.133	**q3**	~ 110
l4	0.073	**q4**	~ -145 a 145
l5	0.06	**q5**	-80 a 60
l6	0	**q6**	-150 a 150

Tabla 2.1. Valores utilizados para la parametrización.

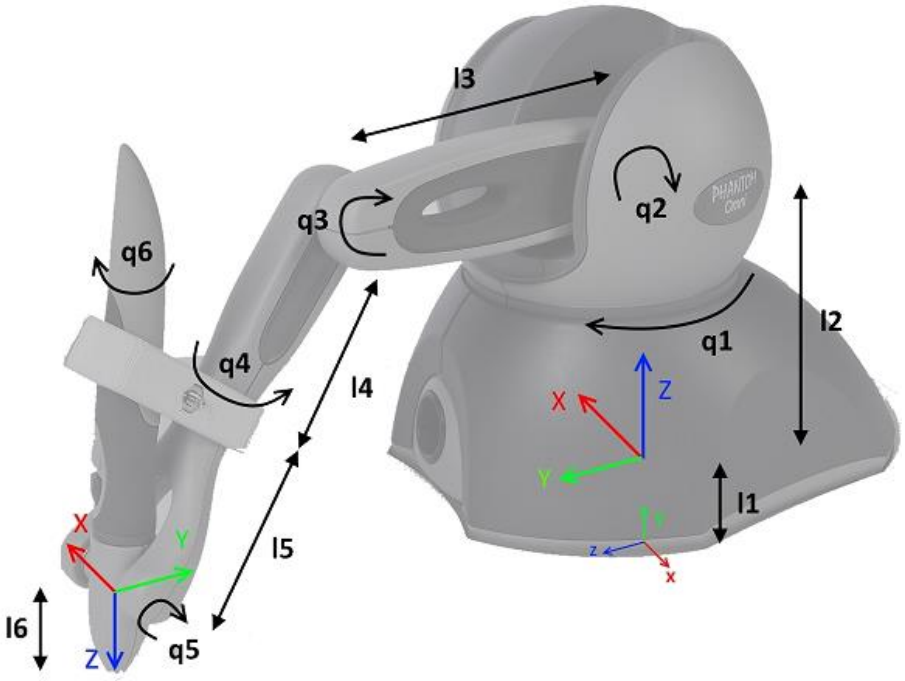


Figura 2.5. Esquema utilizado para la parametrización. Fuente: Modificado de [16].

Con lo manifestado anteriormente, los parámetros de Denavit-Hartenberg son los mostrados en la Tabla 2.2.

	Theta (ϴ)	**d**	**a**	**Alpha (α)**
1	180	l1	0	0
2	-q1 + 90°	l2	0	-90°
3	q2 + 180°	0	l3	0
4	q3 – q2	0	0	90°
5	-q4	l4 + l5	0	-90°
6	q5 – 90°	0	0	90°
7	q6 + 90°	l6	0	0

Tabla 2.2. Parámetros de Denavit-Hartenberg para el Phantom Omni.

De tal modo que, para obtener la posición y ubicación de cualquier articulación, incluido el final, basta con aplicar sucesivamente la transformación homogénea descrita en la ecuación (1), hasta llegar al punto deseado.

$$T = \begin{bmatrix} \cos(\theta) & -\cos(\alpha) * \sin(\theta) & \sin(\alpha) * \sin(\theta) & a * \cos(\theta) \\ \sin(\theta) & \cos(\alpha) * \cos(\theta) & -\sin(\alpha) * \cos(\theta) & a * \sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

2.1.3. Comprobación usando MATLAB

Para verificar el modelo obtenido en el apartado anterior, se hizo uso del software *MATLAB*, debido a la facilidad que ofrece para operar con matrices, además se diseñó una interfaz gráfica para que resulte más cómodo la introducción de ángulos y la visualización del *Phantom*. En la Figura 2.6 se observa el programa en ejecución, en el cual se dispone de campos de texto para introducir los ángulos de cada articulación, y se muestra en un espacio 3D un diagrama del dispositivo en respuesta a los ángulos introducidos, así como también los marcos de referencia con la convención XYZ – RGB (Rojo, Verde y Azul) para los ejes, tanto para el origen como para la punta del lápiz.

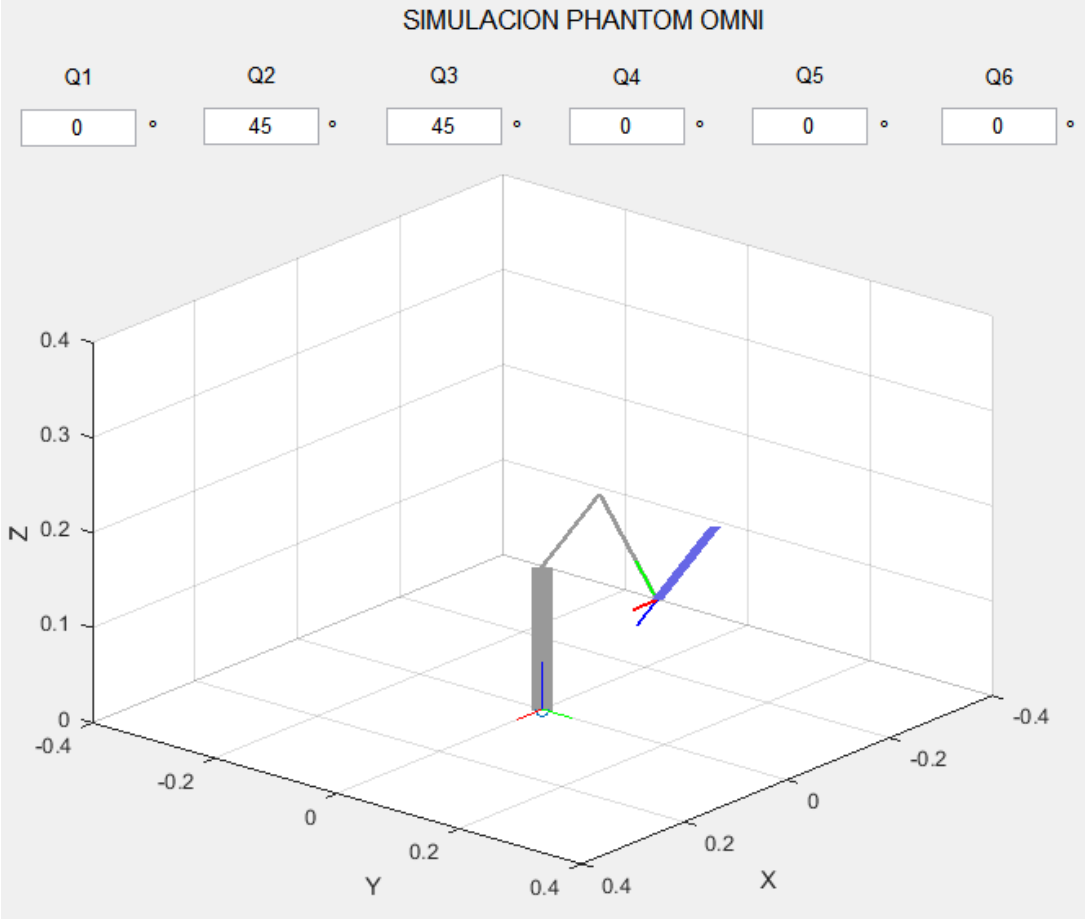


Figura 2.6. Interfaz gráfica en *MATLAB* para simular el *Phantom Omni*.

Con el programa implementado y mediante comparación entre el diagrama mostrado por *MATLAB* y el dispositivo háptico real con los mismos ángulos, se corroboró que los parámetros de Denavit-Hartenberg son correctos.

2.2. Brazo robótico

El manipulador robótico empleado para este proyecto es la versión *MICO2* de la empresa *KINOVA*, junto con una pinza de tres dedos, tal y como se ve en la imagen ofrecida por el fabricante en la Figura 2.7, es un brazo colaborativo, principalmente usado para brindar asistencia en entornos cotidianos a personas con algún impedimento físico o motriz.

Figura 2.7. Brazo robótico *Kinova MICO2*. Fuente: [17].

2.2.1. Características del brazo Kinova MICO2

Como se dijo en el apartado anterior, el marco de referencia del brazo robótico es diferente al usado por el dispositivo háptico, en la Figura 2.8 se muestra los ejes que emplea el *MICO2*, y ya que será el mecanismo que vamos a teleoperar, se adaptó el *Phantom* para coincidir con éste. Además, se puede observar la notación para la orientación, siendo el ángulo Theta X el que lo orienta arriba/abajo, el ángulo Theta Y alrededor de los dedos, y finalmente Theta Z que es el giro circular de la muñeca.

De las características expuestas en la Figura 2.9, las más sobresalientes para este proyecto son: alcance máximo de 70 centímetros, peso de carga de aproximadamente 1 kilogramo, y el sistema de control que ofrece, cartesiano o angular.

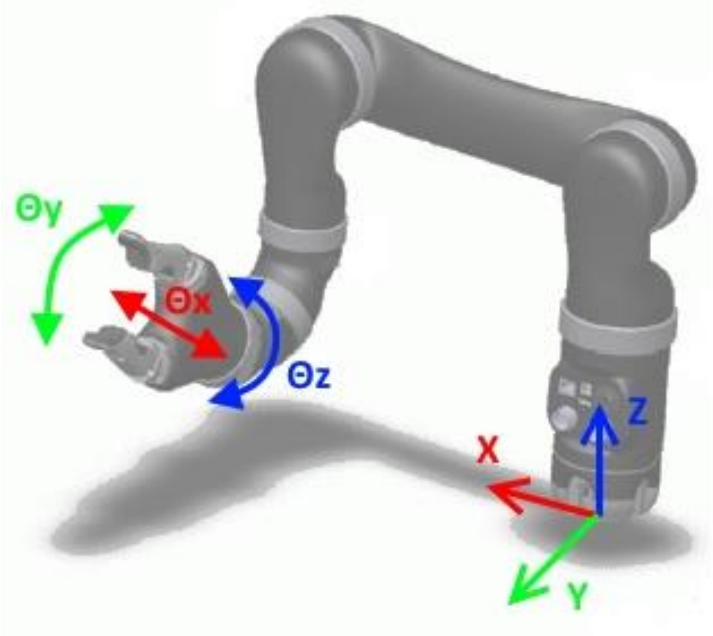


Figura 2.8. Ejes de referencia para *MICO2*. Fuente: Modificado de [18].

GENERAL				
		NO GRIPPER	2 FINGERS (KG-2)	3 FINGERS (KG-3)
Total weight		4.6 kg	5.2 kg	5.4 kg
Payload capabilities	Mid-range continuous	2.1 kg	1.3 kg	1.1 kg
	Full-reach peak/temporary	1.5 kg	0.8 kg	0.6 kg
Materials	Links	Reinforced plastic		
	Actuators	Aluminum		
Maximum reach		70 cm		
Joint range after start-up *(sotware limitation)*		±27.7 turns		
Maximum linear arm speed		20 cm/s		
Power supply voltage		18 to 29 VDC		
Peak power		100 W		
Average power	Operating mode	25 W		
	Standby mode	5 W		
Communication protocol		RS-485		
Communication cables		20 pins flat flex cable		
Expansion pins		2 *(on communication bus)*		
Water resistance		IPX2		
Operating temperature		-10 °C to 40 °C		

CONTROLLER		
Ports	Joystick	1 Mbps Canbus
	Power supply	18 to 29 VDC
	USB 2.0 (API)	12 Mbps
	Ethernet (API)	100 Mbps
Control system frequency	High level (API)	100 Hz
	Low level (API)	500 Hz
CPU		360 MHz
SDK	APIs	High and low level
	Compatibility	Windows, Linux Ubuntu & ROS
	Port	USB 2.0, Ethernet
	Programming languages	C++
Control		Force, cartesian & angular

Figura 2.9. Especificaciones del brazo *MICO2*. Fuente: [17].

2.3. Integración entre dispositivos

Una vez expuestas las características principales de cada dispositivo a utilizar, así como sus ejes de referencia, parametrización y métodos de control, se dispone a integrar ambos artefactos en el mismo espacio de trabajo. Si bien se permite el control articular del brazo robótico, y se puede obtener los ángulos de cada articulación del mecanismo háptico, un control de este tipo no sería adecuado debido a que ambos artefactos presentan distintos modelos cinemáticos, a pesar de que presenten igual número de grados de libertad. Debido a esto, se decide usar el espacio cartesiano XYZ para unificar los espacios de trabajo.

Mediante el modelado por Denavit-Hartenberg para el *Phantom*, se hizo coincidir los ejes de referencia, en consecuencia, ya se está trabajando sobre el mismo espacio de trabajo, con la diferencia de que el área útil del dispositivo háptico es menor a la del *MICO2*, lo que conlleva a que deba existir un escalamiento en cada eje. Adicionalmente, hay que tomar en cuenta que el brazo cuenta con una zona de protección alrededor del origen, un cilindro de aproximadamente 15 centímetros de radio, estos conceptos se ilustran en la Figura 2.10. Finalmente, se hace notar que el espacio operativo del brazo se encuentra en el eje Y negativo, ya que hacia el eje positivo se encuentran los puertos de conexión y, por tanto, los cables.

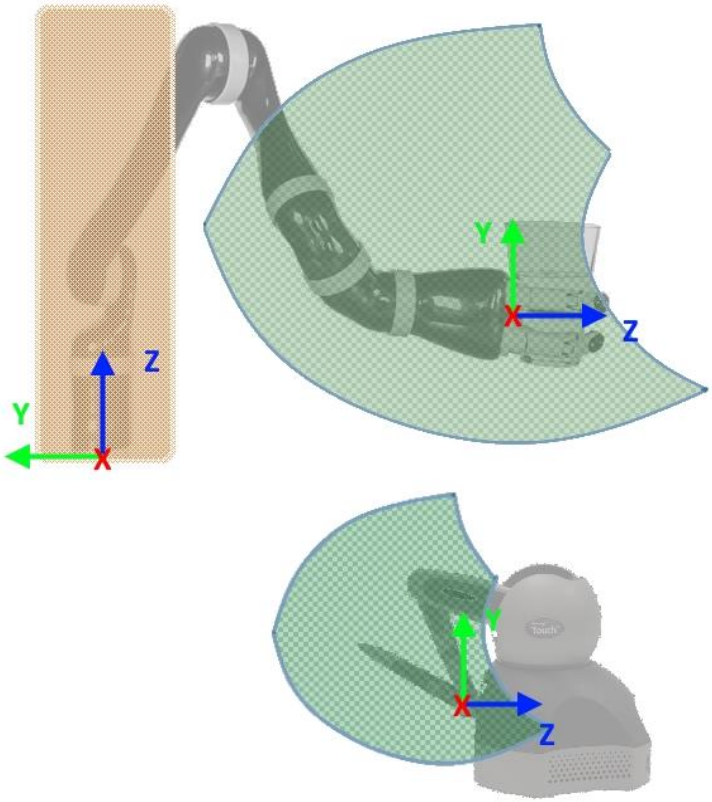


Figura 2.10. Espacio de trabajo *Phantom- MICO*.

En virtud de lo expuesto anteriormente, la formulación matemática necesaria para ajustar la posición del brazo robótico respecto al dispositivo háptico se reduce a utilizar la ecuación de la recta, como se presenta a continuación.

$$X_M = b_x + m_x * X_P$$

$$Y_M = b_y + m_Y * (Y_P - 0.27) \tag{2}$$

$$Z_M = b_z + m_Z * Z_P$$

Donde:

El subíndice *M* corresponde al *MICO2.*

El subíndice *P* corresponde al *Phantom*.

En la ecuación de la componente Y hay un ligero cambio debido a que se está trabajando en sentido negativo, como se explicó antes. El valor de 0.27 corresponde a los 270 milímetros que es el máximo alcanzable por el *Phantom* en el eje Y.

El control sobre el manipulador, además de posición, incluye orientación, mediante los ángulos Theta X, Theta Y, Theta Z; cuyo formato se encuentra en ángulos de Euler XYZ, los mismos que se pueden obtener de la submatriz de rotación de la matriz de transformación homogénea, como se indica a continuación.

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3}$$

$$R = \begin{bmatrix} \mathrm{c}(\theta_y)\mathrm{c}(\theta_z) & -\mathrm{c}(\theta_y)\mathrm{s}(\theta_z) & \mathrm{s}(\theta_y) \\ \mathrm{c}(\theta_x)\,\mathrm{s}(\theta_z) + \mathrm{c}(\theta_z)\,\mathrm{s}(\theta_x)\,\mathrm{s}(\theta_y) & \mathrm{c}(\theta_x)\,\mathrm{c}(\theta_z) - \mathrm{s}(\theta_x)\,\mathrm{s}(\theta_y)\,\mathrm{s}(\theta_z) & -\mathrm{c}(\theta_y)\mathrm{s}(\theta_x) \\ \mathrm{s}(\theta_x)\,\mathrm{s}(\theta_z) - \mathrm{c}(\theta_x)\,\mathrm{c}(\theta_z)\,\mathrm{s}(\theta_y) & \mathrm{c}(\theta_z)\,\mathrm{s}(\theta_x) + \mathrm{c}(\theta_x)\,\mathrm{s}(\theta_y)\,\mathrm{s}(\theta_z) & \mathrm{c}(\theta_x)\mathrm{c}(\theta_y) \end{bmatrix}$$

Donde:

c representa a la función coseno

s representa la función seno

R es la submatriz de rotación de la transformación homogénea

De tal modo que se puede hallar los ángulos como sigue:

$$\theta_x = atan2\left(\frac{-R_{23}}{R_{33}}\right)$$

$$\theta_y = atan2\left(\frac{-R_{13}}{\sqrt{(R_{11})^2 + (R_{12})^2}}\right) \tag{4}$$

$$\theta_z = atan2\left(\frac{-R_{12}}{R_{11}}\right)$$

Donde:

atan2 representa la función inversa de la tangente en cuatro cuadrantes

Nótese que existe una singularidad, denominada *gimbal lock* (bloqueo del cardán) cuando el valor absoluto del elemento *R13* es igual a uno, ya que eso significa que *sin(θy)* es la unidad, por consiguiente, *cos(θy)* es nulo, lo que obliga a los elementos *R11, R12, R23* y *R33* a valer cero, y esto causa indeterminación en las ecuaciones para Theta X y Theta Z antes descritas.

Para ésta situación en particular, habrá que determinar los ángulos de la siguiente forma. Considerando que la matriz homogénea de rotación representa una terna ortonormal, así pues, cada elemento constituye entonces el coseno del ángulo entre los respectivos ejes que representan [19], esto significa que si el elemento *R13* tiene como magnitud la unidad, el ángulo entre el eje X y el eje Z de ambos sistemas de referencia es nulo, es decir, están alineados; y el ángulo en Theta X será el formado por los ejes Y, esto es, el elemento *R22*, deduciendo entonces que:

$$\theta_x = \text{acos}(R_{22})$$

$$\theta_z = 0$$

Donde:

El signo de *θx* estará determinado por el signo del elemento *R32*.

θy se obtiene con la misma ecuación mostrada en (4).

2.3.1. Simulación con MATLAB y V-REP

Antes de implementar cualquier proyecto, y de ser posible, es conveniente simularlo antes, para detectar cualquier fallo u omisión que se haya cometido en el desarrollo del mismo. En este caso se hará uso del programa en *MATLAB* elaborado en la sección anterior para el *Phantom Omni* modificado para que tenga comunicación con el *software V-REP* mediante las *API* remotas que ofrece el mismo, y además realice el escalamiento necesario para el espacio de trabajo del brazo *MICO2*. En la Figura 2.11 se muestran los cambios realizados, mostrando el valor de posición y los ángulos para cada dispositivo, varias opciones de vista, y un botón para iniciar la comunicación con *V-REP*.

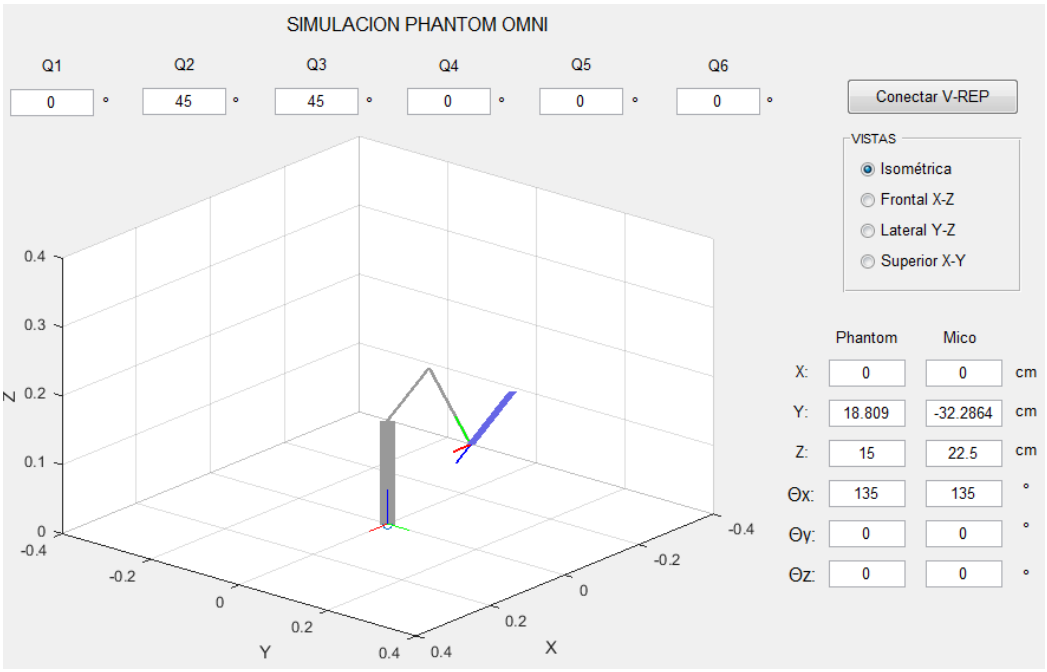


Figura 2.11. Interfaz gráfica en *MATLAB* para conectar con *V-REP*.

En la Figura 2.12 se exhibe la escena en *V-REP*, se ha configurado cada articulación para que trabaje en modo de cinemática inversa; para que esto funcione es necesario describir el punto origen y punto final mediante objetos tipo *dummy*, se ha creado el objeto *Dummy_tip* unido a la mano del brazo (origen), y *Dummy_target* unido al origen del *MICO* (destino), en consecuencia, la cinemática inversa incluye todas las articulaciones comprendidas entre estos objetos *dummy*, que coincide con los seis grados de libertad del brazo robótico.

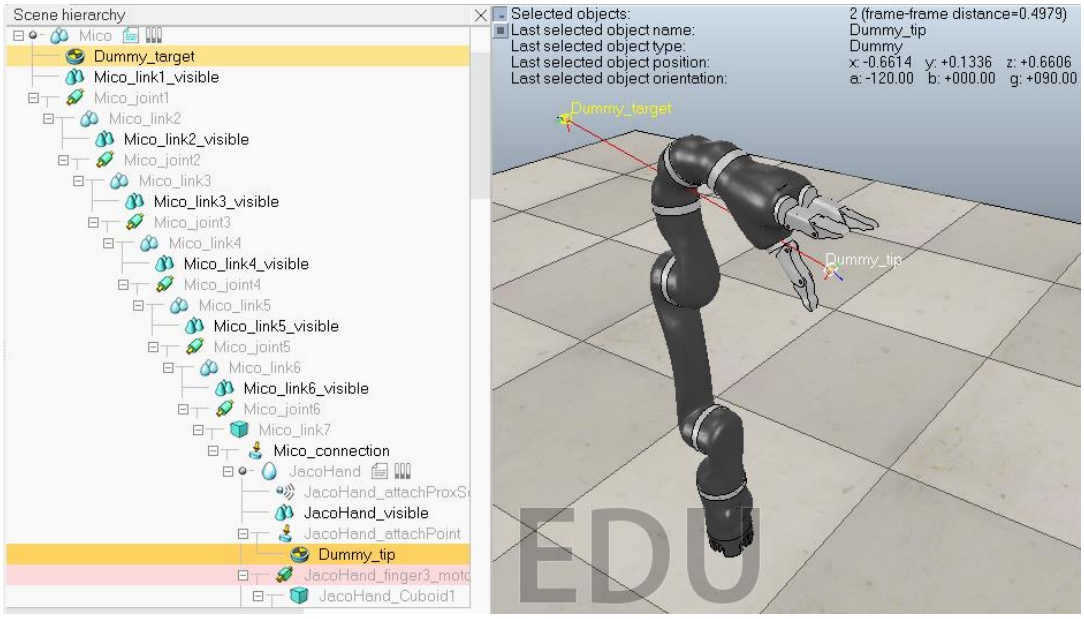


Figura 2.12. Escena en *V-REP*.

En la sección de cálculos, se debe agregar un nuevo grupo cinemático que incorpore los objetos creados anteriormente, en el cual se podrá configurar el método de cálculo, iteraciones, etc. Además de especificar los objetos destino y origen. En la Figura 2.13 se indica la configuración utilizada en este proyecto.

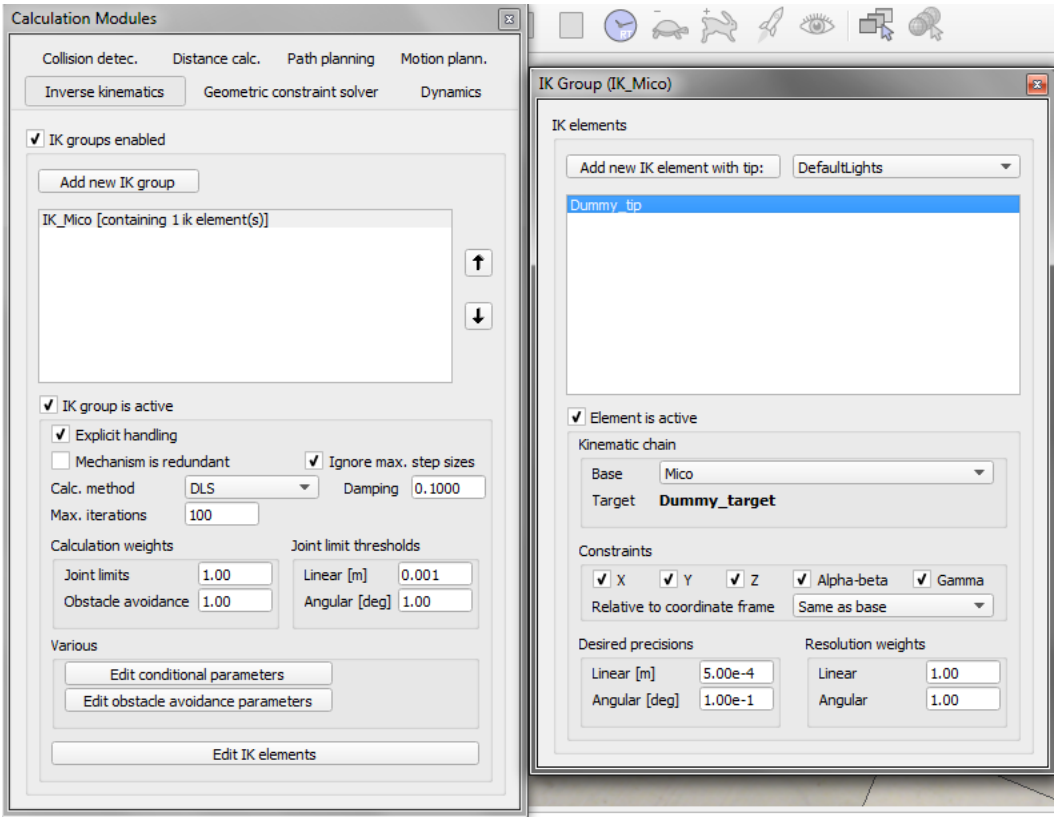


Figura 2.13. Configuración de cinemática inversa para el *MICO* en *V-REP*.

Con lo expuesto previamente, el brazo robótico hará coincidir el *Dummy_tip* con el *Dummy_target*, tanto en posición como en orientación, calculando y posicionando las articulaciones en los ángulos necesarios. De manera que, desde *MATLAB* se modificará la posición y orientación del objeto *Dummy_target* y el software *V-REP* realizará el control necesario para alcanzar este punto.

Cabe recordar que al abrir el software *V-REP* es necesario iniciar la *API* remota para que *MATLAB* pueda establecer comunicación. Para esto, se debe ejecutar la orden mostrada a continuación en la línea de comandos *Input Lua Code*.

```
simRemoteApi.start(23000,1300,false,true)
```

Con la simulación realizada, se ha podido verificar que el brazo robótico es capaz de conseguir cualquier posición y orientación dentro del espacio de trabajo del dispositivo háptico, además de corroborar que los ajustes matemáticos necesarios para la teleoperación son correctos, con éstos resultados satisfactorios, de manera teórica se ha conseguido que el *MICO* replique los movimientos del *Phantom*, en consecuencia, el siguiente paso es implementar los controles necesarios para ubicar y posicionar el brazo robótico en la práctica.

En la Figura 2.14 se muestra uno de los resultados obtenidos al simular los dispositivos maestro y esclavo para una posición y ubicación determinados, en la cual se exhiben diferentes vistas para mayor comprensión, cabe destacar como en cada plano, la orientación coincide precisamente con la del lápiz del dispositivo háptico; en cuanto a ubicación se han utilizado las fórmulas expuestas en (2), resultando para este caso en particular como se indica a continuación.

$$X_M = 1.8 * X_P$$

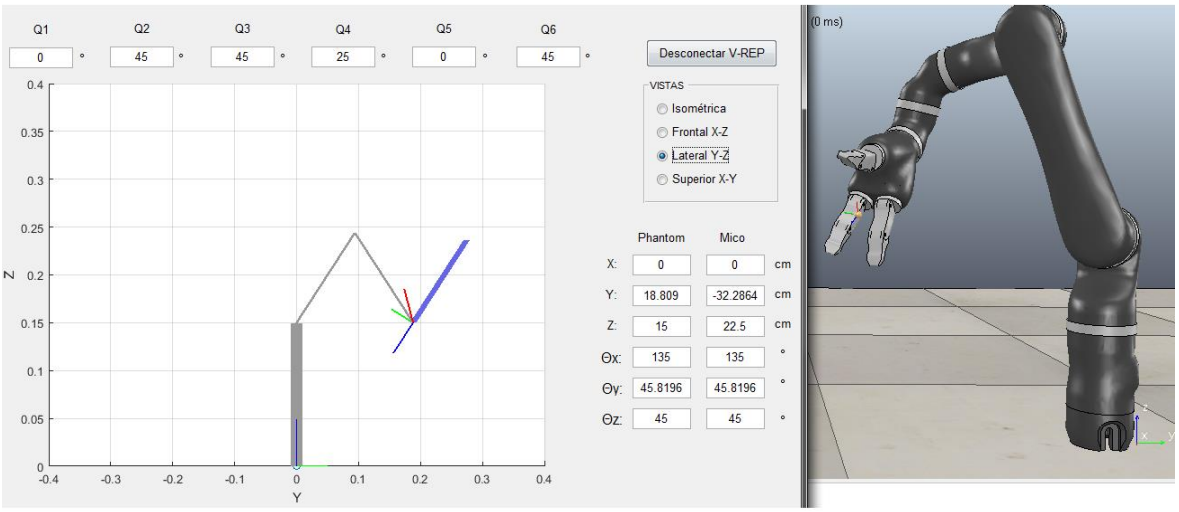
$$Y_M = -0.2 + 1.5 * (Y_P - 0.27) \tag{5}$$

$$Z_M = 1.5 * Z_P$$

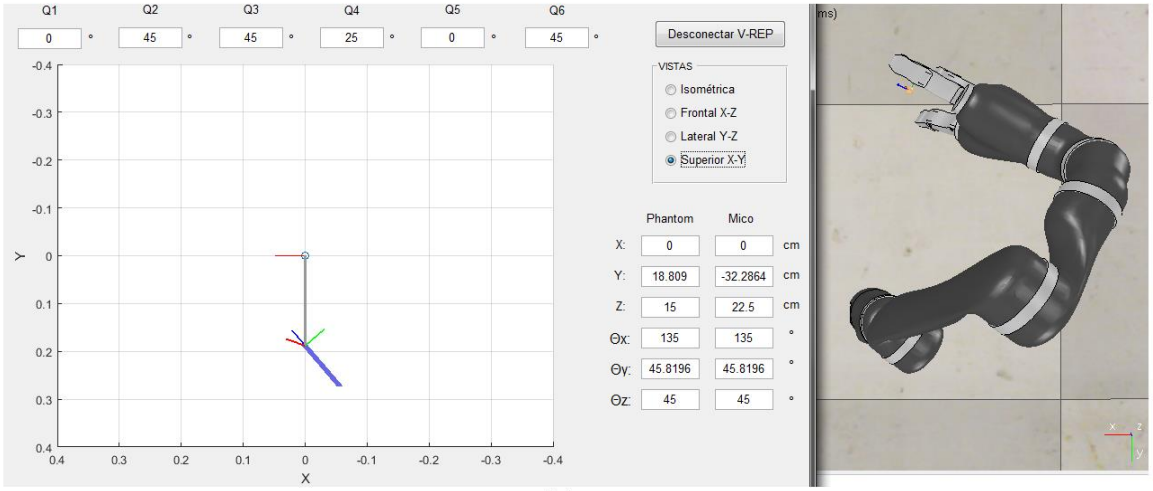
Donde:

El subíndice *M* corresponde al *MICO2.*

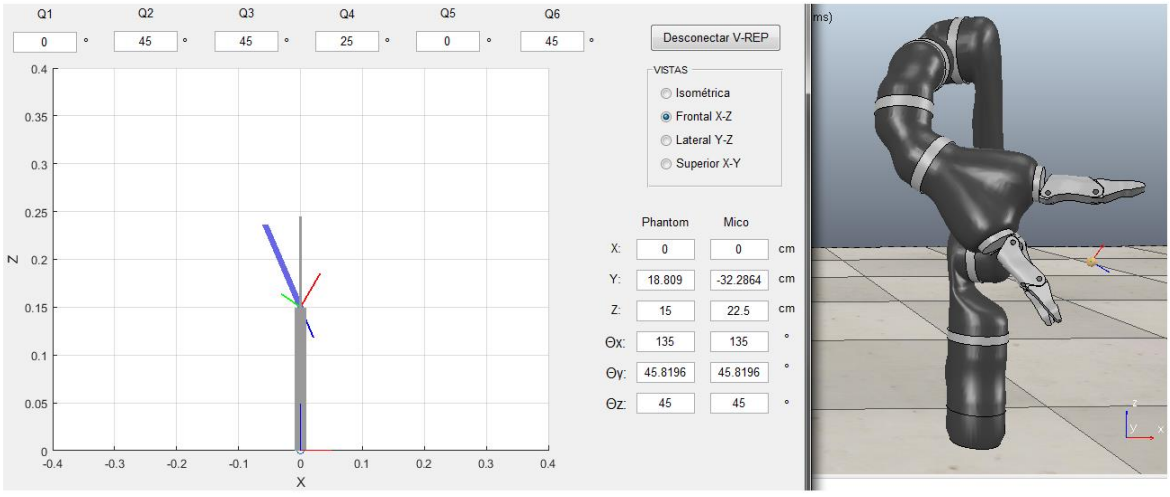
El subíndice *P* corresponde al *Phantom*.



(a)



(b)



(c)

Figura 2.14. Simulación mediante *MATLAB* y *V-REP*. (a) Vista lateral Y-Z, (b) Vista superior X-Y y (c)Vista frontal X-Z.

2.3.2. Control en posición cartesiana

Como se mencionó en las características del brazo robótico, *Kinova* ofrece mediante su *SDK* la posibilidad de controlar la traslación y orientación del extremo final del *MICO* a través del modo de posición cartesiana, esto es, indicarle la ubicación mediante las coordenadas (X,Y,Z) en metros, y la orientación con los ángulos (Theta X,Theta Y,Theta Z) en radianes; es decir, de la misma manera en que se realizó la simulación, en consecuencia, el controlador propio del robot se encargará de realizar la trayectoria necesaria para alcanzar dicho punto. Es lógico entonces pensar en este modo como primera opción para comandar el brazo robótico, ya que resulta más fácil e intuitiva de implementar, por consiguiente, será el primer caso que se analizará.

En la Figura 2.15 se presenta los resultados de una trayectoria empleando este modo de mando, se puede observar cada componente cartesiana en milímetros a través del tiempo discreto, usando un tiempo de muestreo de aproximadamente doscientos milisegundos.

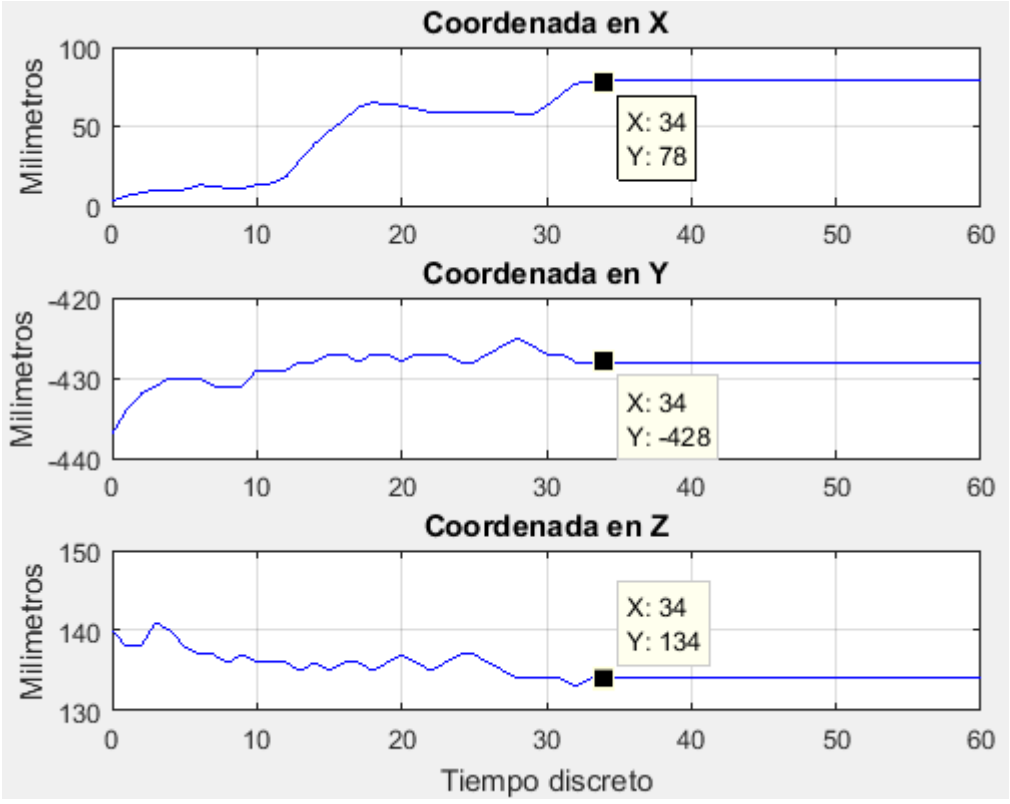


Figura 2.15. Traslación del brazo robótico en el tiempo usando el modo de posición cartesiana.

En la imagen antes mencionada, también se ha añadido marcadores que indican la posición en el tiempo con su valor en milímetros respectivo, es notable que para tiempos anteriores existe variación y por tanto movimiento, no así para tiempos superiores donde se observa un valor constante en las tres componentes de traslación, que se traduce en que el brazo robótico está totalmente inmóvil, a pesar de que se le siga enviando trayectorias diferentes. Esto se debe a que, de acuerdo a la ayuda ofrecida en la documentación de la *API*, las posiciones que se envían al controlador, se almacenan y se procedan mediante un *buffer FIFO*, esto implica que el robot debe alcanzar un punto antes de continuar con el siguiente, sin importar el tiempo que pueda tardar, por lo cual, al enviarle trayectorias continuamente en cortos períodos de tiempo, éste se bloquea y no responde a las peticiones de movimiento. Entre las posibles soluciones tenemos: aumentar el tiempo de muestreo del *Phantom* para no saturar el *buffer* y conseguir así que pueda ejecutar cada movimiento, o verificar que se ha llegado al último punto antes de enviar nueva información. Pero esto conlleva, en el primer caso, a sacrificar la percepción de tiempo real, ya que, a mayor tiempo de muestreo, más lenta será la respuesta, notando retardos entre el movimiento del dispositivo maestro y la respuesta en el brazo; y en el segundo caso, involucra ignorar los movimientos intermedios hasta que el robot finalice la trayectoria actual, causando una teleoperación poco efectiva.

Con lo descrito anteriormente se puede concluir que este modo de operación no es el más adecuado para aplicaciones de seguimiento en tiempo real, en cambio, resulta bastante efectiva para tareas cuyas trayectorias estén definidas previamente, o no se necesite tener respuesta inmediata, por ejemplo, programación de tareas repetitivas como *pick and place*, etc.

2.3.3. Control en velocidad cartesiana

Otro método que ofrece las herramientas para desarrollador, es el modo de velocidad cartesiana, el cual aún ofrece la ventaja de poder trabajar en un espacio cartesiano, en vez de articular, pero con la diferencia de que el controlador integrado se encarga de mover el extremo final del robot a la velocidad indicada en cada eje, esto permite un control de dirección mas no de posición, en consecuencia, para llegar a una posición deseada habrá que implementar un controlador externo que cumpla este cometido; en contraste, este método soluciona el problema del modo de posición cartesiana para aplicaciones de seguimiento, ya que para su funcionamiento requiere una constante actualización de velocidad, y en el caso de no tener ninguna información, asume que la velocidad de movimiento es nulo y por

tanto, se detiene; dicha tasa de actualización es bastante corta, en el orden de unidades de milisegundos, permitiendo una respuesta en tiempo real ante las órdenes enviadas desde el dispositivo háptico, y evitando que se pierdan datos intermedios durante el muestreo.

En la Figura 2.16 se expone el resultado de utilizar el modo de velocidad cartesiana, con gráficas de cada coordenada en el espacio a lo largo del tiempo, se puede apreciar que existe movimiento para cada instante de muestreo, cabe destacar incluso que el tiempo es mayor respecto al mostrado en el modo de posición cartesiana, y el robot ha respondido adecuadamente durante todo el experimento.

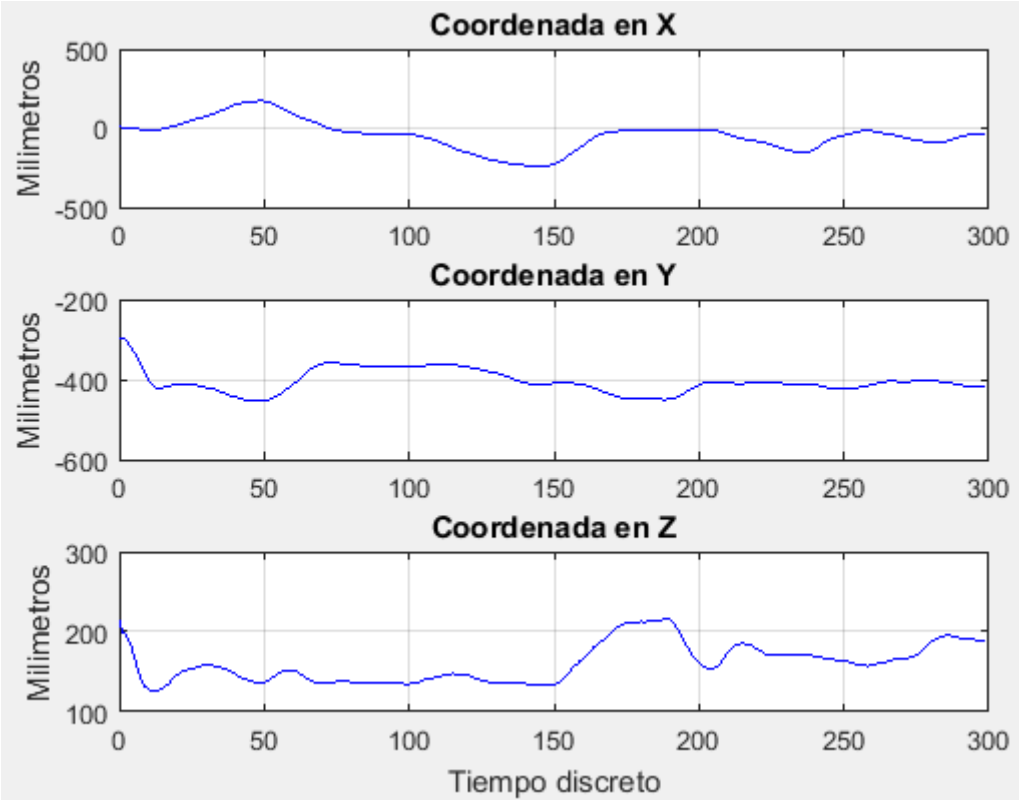


Figura 2.16. Traslación del brazo robótico en el tiempo usando el modo de velocidad cartesiana.

Se puede concluir entonces, que este modo es más adecuado para tareas que demanden respuesta instantánea por parte del efector final, con el añadido de tener que diseñar un controlador aparte si se desea llegar a una posición determinada. Adicionalmente brinda protección ante pérdidas de comunicación, ya que, al no recibir órdenes de movimiento, el robot mantendrá la posición actual.

2.3.4. Filtro IIR de primer orden

Al obtener la lectura de los ángulos del *Phantom* y realizar las transformaciones necesarias para obtener la posición y orientación final, se ha visto necesario implementar un filtro pasobajo, ya que al manipular el dispositivo maestro existen ligeras perturbaciones propias de un manejo manual, como vibraciones o movimientos involuntarios del operador, y cabe recordar que existe un factor de escalamiento para el espacio de trabajo del manipulador, de tal modo que éstos movimientos indeseados se amplificarán en el efector final, debido a lo cual se propone aplicar un filtro para eliminar estos desplazamientos de alta frecuencia, y aumentar así la precisión en la trayectoria. Esto se puede evidenciar en la Figura 2.17, en la cual se presentan las componentes cartesianas de un recorrido aleatorio realizado con el lápiz del *Phantom*, conjuntamente con su respectivo espectro frecuencial mostrado en Hertz, en (a) se alcanza a visualizar vibraciones sobrepuestas a la trayectoria, por ejemplo, alrededor de los dos segundos, y en (b) se logra apreciar que la mayor parte de información se encuentra antes de los cinco Hertz aproximadamente.

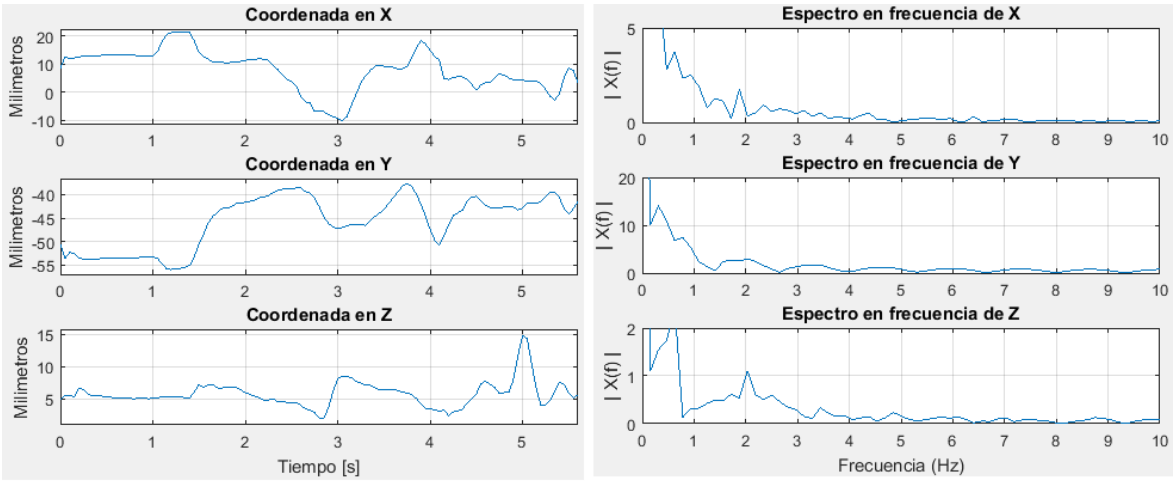


Figura 2.17. Trayectoria medida en el *Phantom* sin filtro. (a) Coordenadas en el tiempo, (b) Espectro frecuencial de las coordenadas.

Con lo detallado anteriormente, se plantea implementar un filtro con respuesta impulsional infinita (*IIR*), debido a que proporciona una pendiente de corte más pronunciada y con menor retardo que un filtro de respuesta finita (FIR) [20], teniendo en cuenta que el ruido a eliminar no es muy prominente, se opta por un filtro de orden uno.

Para poner en práctica el filtro, previamente se debe diseñarlo para cumplir las especificaciones requeridas, y esto conlleva a conocer la formulación matemática; debido a que su implementación se hará por *software*, se está hablando de un filtro digital, por ende, su expresión matemática está en función del tiempo discreto, la misma que se detalla en [21], adaptado para un filtro pasobajo como sigue:

$$y(n) = \sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M-1} b_k x(n-k) \tag{6}$$

Donde:

y representa la salida del filtro

x representa la entrada del filtro

a y *b* son coeficientes que determinan la respuesta del filtro

Para el diseño de un filtro, es más útil el análisis en el dominio frecuencial, esto es, la transformada Z para funciones discretas. De [21] obtenemos la expresión en dominio Z como:

$$H(z) = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{1 + \sum_{k=0}^{N} a_k z^{-k}} \tag{7}$$

Con la expresión antes descrita, y con la ayuda de *MATLAB*, se puede determinar la respuesta del filtro en frecuencia y fase por medio del diagrama de Bode, y con los coeficientes *a* y *b* ajustar la frecuencia de corte, que se concluyó anteriormente rondará los cinco Hertz.

Mediante pruebas con diferentes coeficientes, se decide fijar los valores en *a=0.3* y *b=0.7* obteniendo los resultados expuestos en la Figura 2.18, en el marcador agregado puede verse que la frecuencia de corte es de 4.42 Hertz, y un desfase máximo de diecisiete grados aproximadamente, lo cual es una respuesta adecuada para esta aplicación. En consecuencia, y basándose en la ecuación (6), la implementación del filtro IIR de primer orden en un bucle de muestreo tiene la forma siguiente.

$$Posición\ (x, y, z) = \left(0.3 * Posición_{Anterior}(x, y, z)\right) + \left(0.7 * Posición_{Actual}(x, y, z)\right)$$

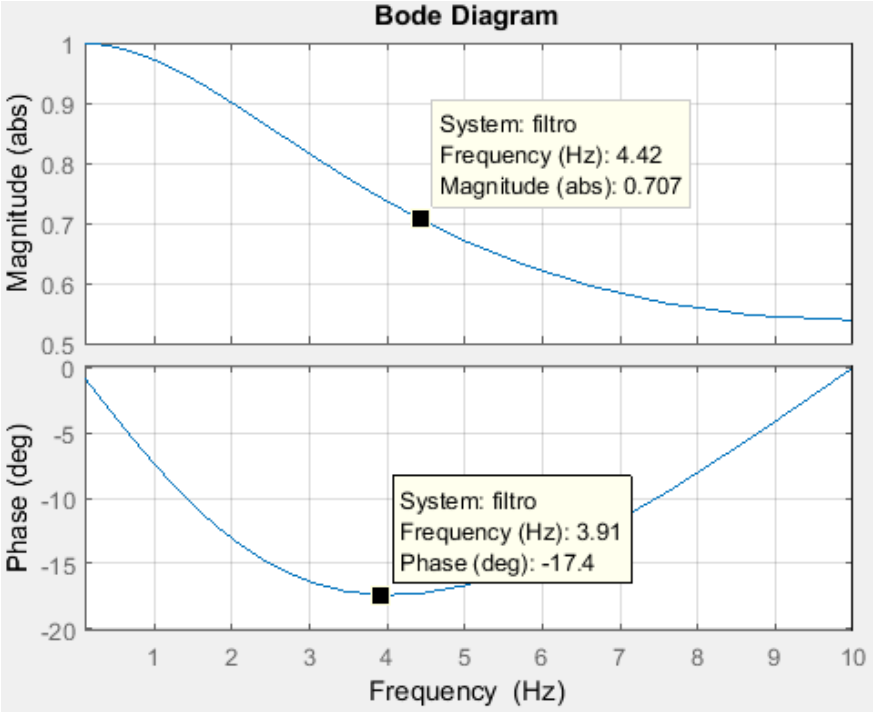


Figura 2.18. Diagrama de Bode para el filtro IIR.

Se comprueba la respuesta efectuando una trayectoria al azar con el filtro implementado, obteniendo los resultados mostrados en la Figura 2.19, en los cuales se observa la atenuación de frecuencias mayores a cinco Hertz, y por tanto, se determina que el comportamiento es el deseado.

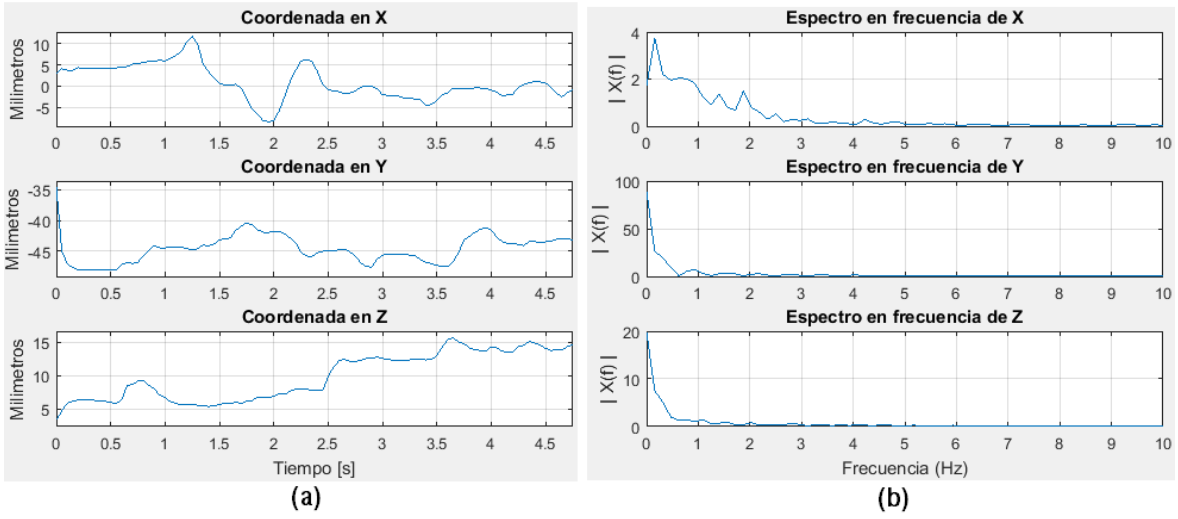


Figura 2.19. Trayectoria medida en el *Phantom* con filtro. (a) Coordenadas en el tiempo, (b) Espectro frecuencial de las coordenadas.

Utilizando el mismo procedimiento antes expuesto, se diseñó un filtro de similares características para la realimentación de fuerza hacia el dispositivo háptico, debido a que el brazo robótico genera perturbaciones considerables

mientras se desplaza, como se puede advertir en la Figura 2.20, la cual muestra las fuerzas medidas sin filtro en el efector a lo largo de una trayectoria.

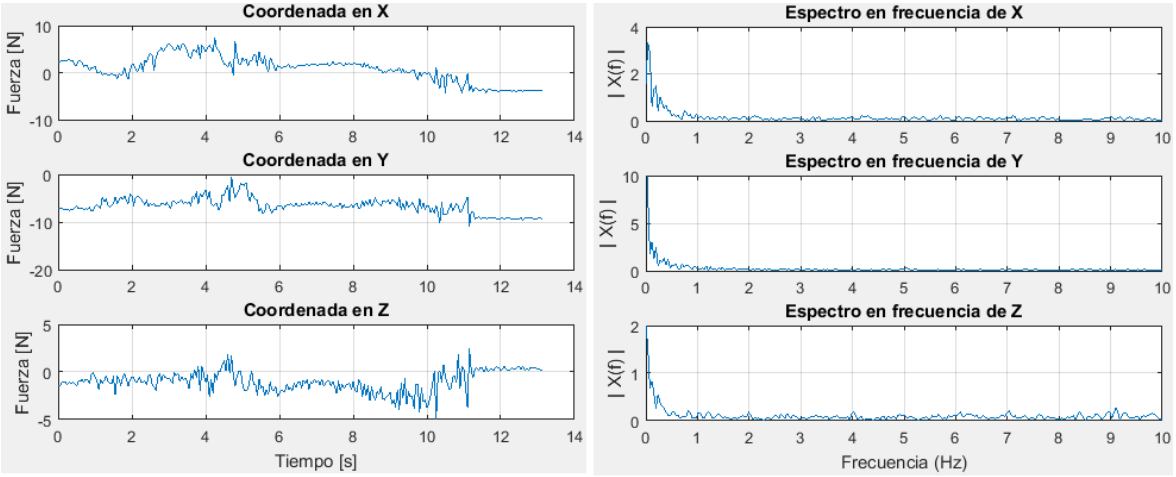


Figura 2.20. Fuerzas medidas en el brazo robótico sin filtro. (a) Coordenadas en el tiempo, (b) Espectro frecuencial de las coordenadas.

Puesto que existe mayor ruido en la información de fuerza, se decide por establecer los coeficientes en: *a=0.4* y *b=0.6*; con lo cual, la expresión para implementar el filtro de realimentación de fuerza, es el indicado a continuación.

$$Fuerza\ (x,y,z) = \left(0.4 * Fuerza_{Anterior}(x,y,z)\right) + \left(0.6 * Fuerza_{Actual}(x,y,z)\right)$$

El resultado obtenido es el expuesto en la Figura 2.21, es notable la reducción de ruido que se ha logrado, validando de esta manera el filtro aplicado.

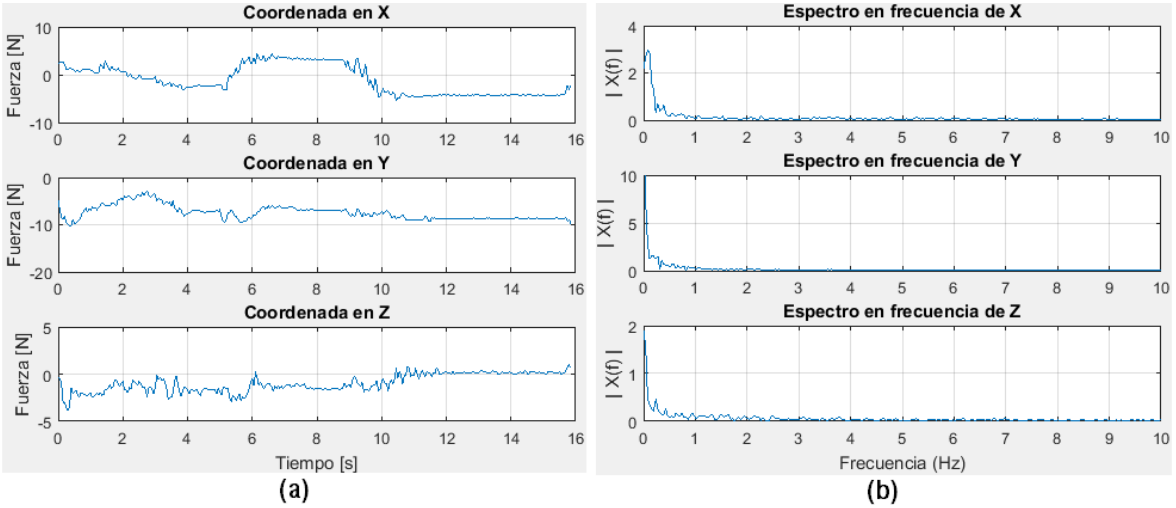


Figura 2.21. Fuerzas medidas en el brazo robótico con filtro. (a) Coordenadas en el tiempo, (b) Espectro frecuencial de las coordenadas.

2.3.5. Control Proporcional

Como se determinó en secciones anteriores de este documento, la alternativa más acertada y, por tanto, la que se va a utilizar para esta investigación en cuanto al modo de operación del brazo robótico, es la opción de velocidad cartesiana, la cual requiere de un control externo para posicionar y orientar el extremo final del robot en el espacio. Intuitivamente se puede llegar a la siguiente solución: enviar el efector a la posición y orientación deseada a cierta velocidad e ir disminuyendo a medida que se aproxima a la coordenada requerida hasta detenerse por completo al lograr el cometido, lo que coincide precisamente con la descripción de un controlador proporcional. Es pertinente rememorar que una trayectoria en velocidad está definida por las componentes (x,y,z) para traslación y (ϴx,ϴy,ϴz) para orientación, por tanto, el control se debe aplicar en forma de vector, como se indica en la expresión a continuación.

$$V = Kp \cdot (Posición_{deseada} - Posición_{actual}) \tag{8}$$

Donde:

V es el vector 1x6 de velocidades a aplicar

Kp es el vector 1x6 de ganancias proporcionales

Posición es el vector 1x6 de posición

. indica una multiplicación componente a componente.

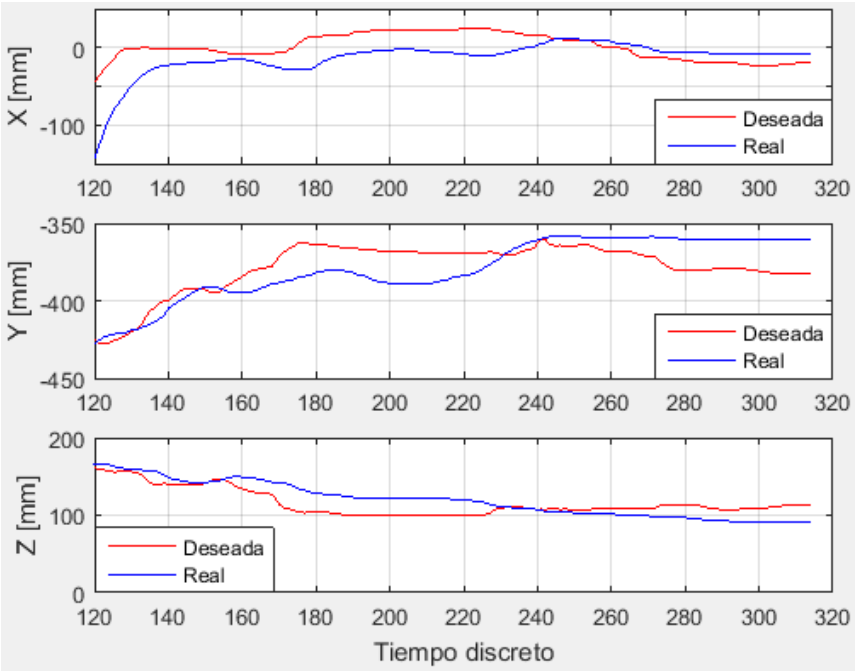


Figura 2.22. Control proporcional aplicado a la traslación del brazo robótico.

En la Figura 2.22 se presentan los resultados al implementar este controlador, con una ganancia *Kp=3.5* para la traslación, cabe mencionar que se obtuvo una respuesta bastante aceptable, al menos de forma visual en la práctica, pero como se percibe en las gráficas, existe error remanente a lo largo de toda la trayectoria, lo cual es propio de un controlador de este tipo, denominándose error en estado estable o permanente, la solución a esto se presenta en el siguiente apartado.

2.3.6. Control Proporcional – Integral

Del mismo modo que se dedujo el control anterior, la solución para eliminar el error en régimen permanente es bastante práctica, incrementar la velocidad de movimiento si a medida que el tiempo avanza, no se ha logrado llegar a la posición pretendida, o dicho matemáticamente, agregar una etapa integral al controlador. Con lo cual, la expresión requerida para implementar dicho controlador, es el que sigue:

$$V = Kp \cdot \left(Error + Ki \cdot \sum_{n=0}^{\infty} Error \right) \tag{9}$$

Donde:

V es el vector 1x6 de velocidades a aplicar

Kp es el vector 1x6 de ganancias proporcionales

Error es el vector 1x6 con las diferencias entre posición deseada y actual

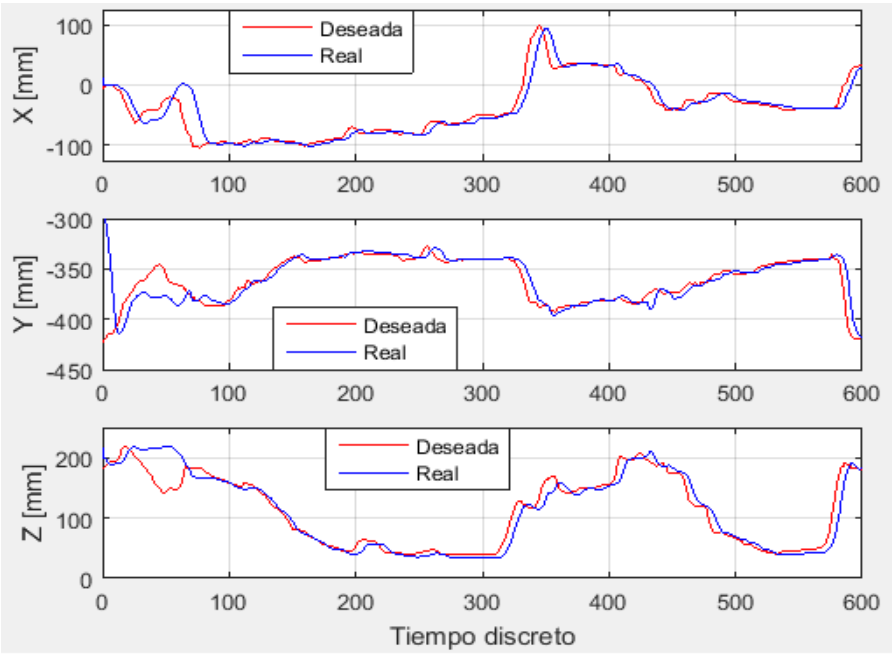
Ki es el vector 1x6 de ganancias integrales

. indica una multiplicación componente a componente

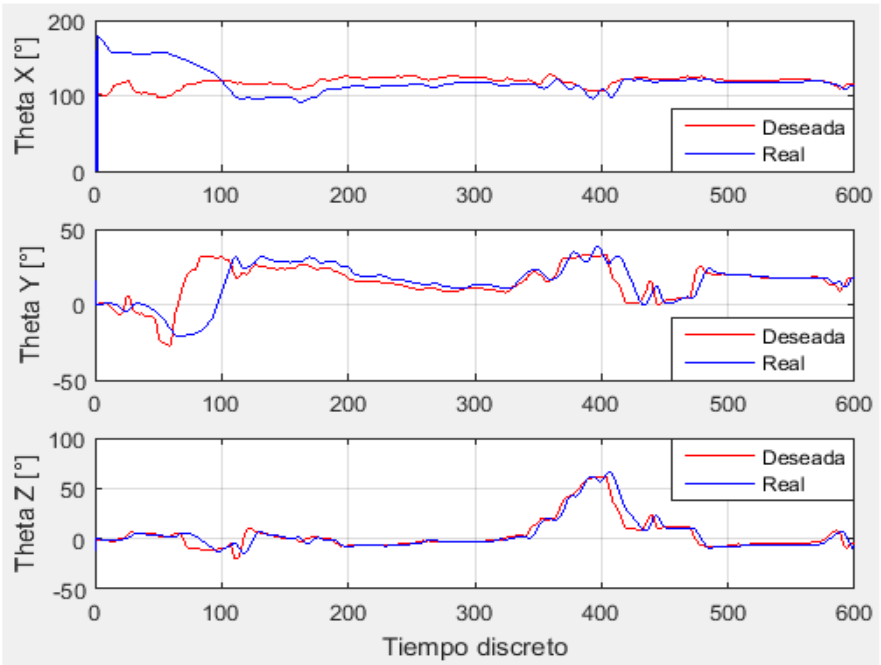
Es pertinente aclarar que la formulación matemática correcta para la acción integral incluye una dependencia al tiempo de muestreo, y dado que en esta aplicación se trata de una teleoperación, lo que significa retardos no homogéneos en el tiempo, se ha decidido no incluir esta variable, puesto que no es indispensable para su funcionamiento.

Como es notable en la Figura 2.23, el controlador ha mejorado significativamente al agregar la componente integral, el seguimiento de la trayectoria es bastante cercano al deseado, y para aplicaciones en entornos cotidianos, como es el caso de este estudio, este control satisface los requerimientos. En la gráfica mencionada, se ilustran los trayectos tanto para traslación como para orientación, es destacable el hecho de que la orientación requiere un tiempo ligeramente mayor para alcanzar la

consigna, mediante varias pruebas se ha llegado a la conclusión de que es una característica propia del controlador interno del *MICO*; otro rasgo considerable es el desfase que se puede apreciar entre la trayectoria consigna y la real, la cual se debe al retardo en la comunicación y al propio desfase introducido por el filtro. En consecuencia, las ganancias para el controlador son: *Kp = [2.1, 2.1, 2.1, 3.5, 3.5, 3.5]* y *Ki = [0.001, 0.001, 0.001, 0.001, 0.001, 0.001]*.



(a)



(b)

Figura 2.23. Control proporcional – integral aplicado al brazo robótico. (a) Traslación, (b) Orientación.

2.4. Comunicación UDP

En el campo de las telecomunicaciones han surgido diversas tecnologías, con variedad de arquitecturas, protocolos, etc. Pero sin duda, las que mayor interconectividad presentan actualmente, son las redes de datos, siendo capaces de enlazar desde dispositivos a nivel doméstico hasta ofrecer una conexión global mediante Internet, por ello, para una aplicación de teleoperación es la opción más apropiada. Dentro de este marco, hay dos principales propuestas a evaluar: *TCP (Transmission Control Protocol)* y *UDP (User Datagram Protocol)*, la primera asegura la recepción de la información, puesto que verifica su llegada y de no ser así, reenvía la información, lo cual implica una trama de mayor tamaño y mayor latencia en la comunicación, por otro lado, la segunda opción es más rápida, debido a que únicamente envía la información al destinatario, sin verificación de recepción y con tramas más cortas. Por este motivo, el protocolo UDP es el más utilizado para aplicaciones que exigen un flujo constante de datos, en los que prima la baja latencia de la telecomunicación, como el monitoreo y control en tiempo real, debido a que no es necesario asegurar que todos los paquetes lleguen al destinatario, ya que dado el caso de que alguno se pierda, inmediatamente será sustituido (según el tiempo de muestreo) por un nuevo dato con un valor próximo al dato extraviado, siendo inapreciable en el entorno práctico.

El protocolo de comunicación nos permite intercambiar información entre los dispositivos, pero es oportuno recordar que se va a enviar y recibir diferentes cifras, tales como: posición, orientación, fuerza, alarmas, etc. De manera que se debe clasificar los datos antes de ser enviados, para poder identificarlos posteriormente en el receptor, se propone entonces, que cada valor esté precedido por una letra mayúscula que, mediante una convención previamente especificada, indique el tipo de información que representa, tal y como se muestra a continuación.

Cabecera UDP	Letra Mayúscula	Dato en formato texto

Ejemplo

Cabecera UDP	‘X’	“1234”

Lo cual podría representar que el valor 1234 corresponde a la coordenada X de una posición.

2.4.1. Comunicación de *Phantom* a *MICO*

Una vez descrito el formato que se va a utilizar para la segmentación de datos, en la Tabla 2.1 se describen las letras, junto con el tipo de dato y su significado, que se van a utilizar para enviar la información requerida para la teleoperación, siendo el dispositivo háptico el emisor y el brazo robótico el receptor.

Letra	Dato	Observaciones
A	Dedo 1	0 – Sin acción. 1 – Cerrar. 2 – Abrir.
B	Dedo 2	
C	Dedo 3	
S	Salir	1 – para finalizar el programa en el esclavo
U	Theta X	Ángulos de Euler deseados, en milésimas de radián.
V	Theta Y	
W	Theta Z	
X	Coordenada X	Posición deseada, en milímetros.
Y	Coordenada Y	
Z	Coordenada Z	

Tabla 2.3. Formato de trama para enviar información desde el *Phantom Omni* hacia el *MICO2*.

2.4.2. Comunicación de *MICO* a *Phantom*

Del mismo modo que en el apartado anterior, el brazo robótico debe remitir información sobre su estado hacia el dispositivo maestro, donde se encuentra la interfaz de usuario para su monitoreo por parte del operador. Así pues, en la Tabla 2.4 se detallan las letras que serán usadas para seccionar la información que el *MICO* envía a la aplicación principal.

Letra	Dato	Observaciones
A	Alarma Brazo	Sobrecarga de fuerza en el brazo robótico.
B	Alarma Dedos	Sobrecarga de fuerza en los dedos.
F	Coordenada X	Fuerza en el efector, en Newton.
G	Coordenada Y	
H	Coordenada Z	
I	Dedo 1	Posición de los dedos. Sin unidad. 0 – Abierto. 6000 – Cerrado.
J	Dedo 2	
K	Dedo 3	
U	Theta X	Ángulos de Euler actual, en milésimas de radián.
V	Theta Y	
W	Theta Z	
X	Coordenada X	Posición actual, en milímetros.
Y	Coordenada Y	
Z	Coordenada Z	

Tabla 2.4. Formato de trama para enviar información desde el *MICO2* hacia el *Phantom Omni*.

2.5. Interfaz de usuario

Como uno de los objetivos de este trabajo, está el diseñar una interfaz gráfica de usuario para monitorear y controlar el sistema de teleoperación, para ello, se ha utilizado el entorno de desarrollo ofrecido por *Microsoft®*, denominado *Visual Studio Community 2017*, la cual es su versión más reciente hasta la elaboración de esta investigación. Los motivos para la elección de este entorno de desarrollo son los siguientes: es un programa ofrecido gratuitamente para desarrolladores, en su versión *Community*; el sistema operativo *Windows®* está presente en la mayoría de

ordenadores actualmente, lo que ofrece mayor compatibilidad para la aplicación; las funciones *API* ofrecidas por los fabricantes de los dispositivos utilizados son fácilmente integrables a este entorno; y finalmente, *Visual Studio* ofrece herramientas que facilitan el diseño de formularios, lo que reduce el tiempo dedicado a esta tarea, y permite dedicarle mayor recursos al programa de control y teleoperación.

Las librerías y demás funciones ofrecidas como herramientas de desarrollo para el *Phantom Omni* y el *MICO2*, están en lenguaje C++, en consecuencia, los códigos fuente de las aplicaciones utilizarán el mismo lenguaje.

2.5.1. Interfaz en el dispositivo esclavo

Ya que el brazo robótico es el dispositivo teleoperado, no requiere de una interfaz gráfica, puesto que toda la información será presentada en el lado del dispositivo maestro, donde estará la persona a cargo del control. Dicho esto, se plantea realizar una aplicación de consola, para ingresar información básica y visualizar mensajes que puedan ayudar a encontrar problemas, en caso de que los hubiese. En la Figura 2.24 se expone la captura de pantalla del programa en ejecución, como primer punto, la aplicación solicita se ingrese el puerto que utilizará el protocolo UDP, que puede ser cualquiera dentro del rango válido; a continuación se irán imprimiendo los resultados de las operaciones efectuadas, tanto para la comunicación, como para la conexión con el brazo robótico; si todo ha sido ejecutado satisfactoriamente, el mensaje "Esperando conexión…" indicará que el programa está listo para recibir las órdenes del dispositivo maestro.

Figura 2.24. Aplicación de consola para el control del brazo *MICO2*.

2.5.2. Interfaz en el dispositivo maestro

La interfaz de usuario se ha dividido en dos pestañas: la primera dedicada únicamente al monitoreo de ambos dispositivos, y la segunda orientada a la configuración, esto es, modificación de parámetros descritos en las secciones anteriores que alteran el funcionamiento del sistema de teleoperación.

En la Figura 2.25, se observa el programa en ejecución con la ventana de monitoreo activa, como elementos generales se tiene: campos de texto para indicar la dirección IP y puerto del dispositivo teleoperado, un botón para iniciar o detener la comunicación con el manipulador, y en el segmento inferior un cuadro de texto, que hará las funciones de consola, donde se mostrará mensajes relevantes sobre el funcionamiento o errores que existan durante la ejecución del mismo.

Figura 2.25. Ventana de monitoreo en interfaz de usuario.

Dentro de la pestaña de monitoreo, se ha dividido la información de los dispositivos en dos paneles: el primero corresponde al dispositivo háptico, en el cual se muestra la posición dentro del espacio de trabajo del *Phantom* en centímetros, así como sus respectivos ángulos en grados sexagesimales, la fuerza que se está ejerciendo en el lápiz en Newton, y los estados de cada botón, junto con las acciones que realiza cada uno al ser presionado; el segundo panel concierne al brazo robótico, se muestra la posición actual del efector dentro del espacio de trabajo del *MICO* en centímetros y la orientación en grados, la fuerza que está soportando el robot en su extremo final, barras de progreso que ilustran el avance de cada uno de los dedos desde el estado de totalmente abiertos hasta totalmente cerrados, y por último, indicadores de alarma para excesos de fuerza.

La pestaña de configuración se exhibe en la Figura 2.26, se encuentran agrupados como se listan a continuación.

General, contiene parámetros que afectan a la ejecución global del sistema, como: el tiempo de muestreo de los dispositivos, la opción de registrar y exportar los datos de posición y fuerza durante la sesión de teleoperación a un archivo de valores separados por coma (.csv), y la verificación para cerrar el programa en el dispositivo esclavo al concluir la comunicación.

Acciones de los botones, permite asignar diferentes acciones a cada botón del lápiz, tales como: sin acción, abrir 2 dedos, cerrar 2 dedos, abrir 3 dedos, y cerrar 3 dedos.

Escalado de posición para Mico, son los parámetros del ajuste lineal que se realiza para ajustar el espacio de trabajo del *Phantom* al espacio de trabajo del brazo.

Escalado de fuerza para *Phantom*, debido a que el dispositivo háptico es capaz de generar un máximo de 3.3 N., y el brazo puede soportar cantidades muy superiores, se ha incluido este valor de escalado para que la realimentación de fuerza sea ajustada al nivel de sensibilidad que la tarea requiera.

Filtros IIR de primer orden, otorga la flexibilidad de ajustar los filtros usados para la posición y fuerza, cabe recalcar que solo uno de los parámetros es editable, puesto que el otro será la diferencia para alcanzar la unidad, ya que estos filtros no tienen ganancias adicionales.

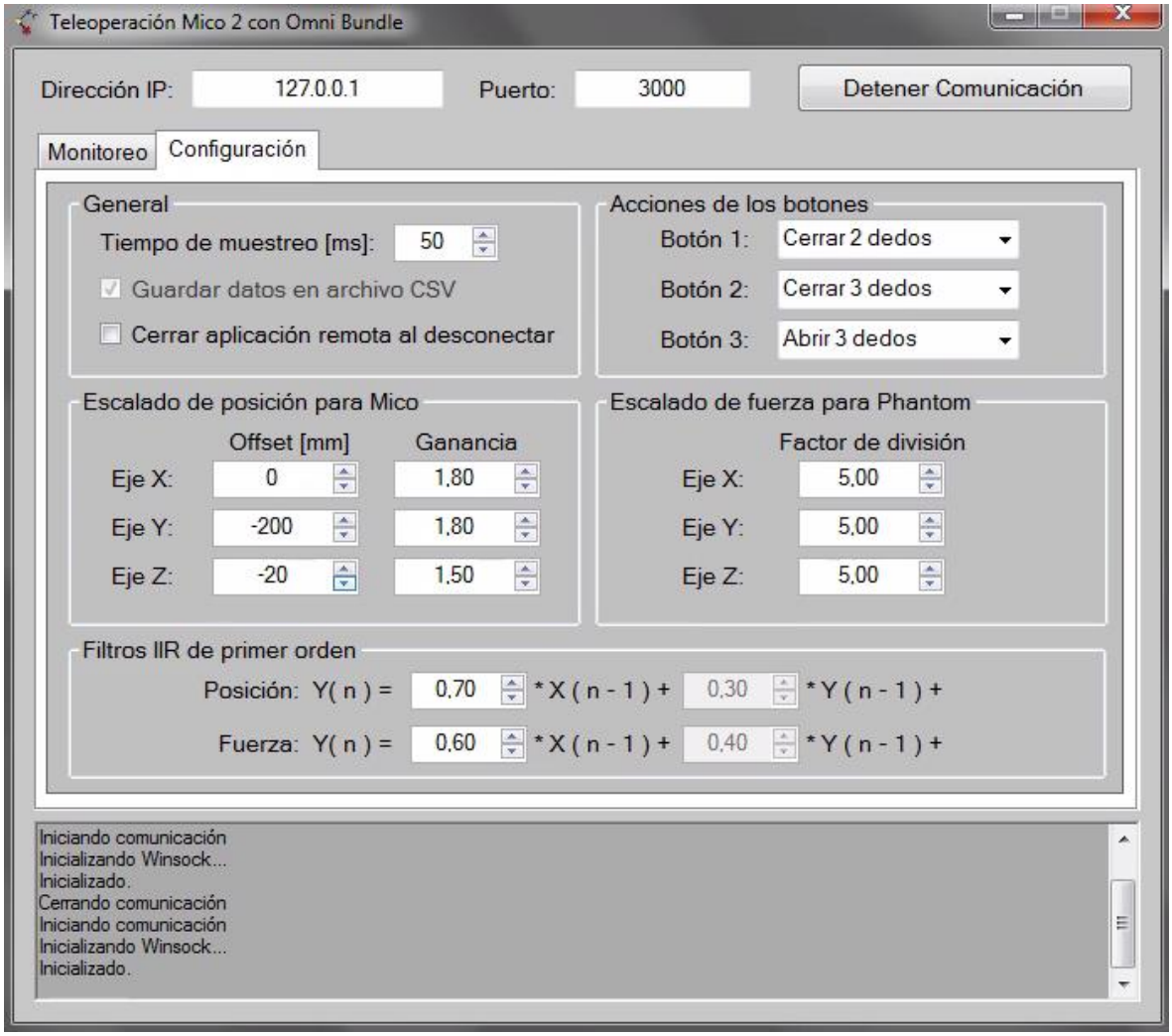


Figura 2.26. Ventana de configuración en interfaz de usuario.

Capítulo 3

Experimentación y resultados

En este apartado se expondrá los experimentos realizados para validar el funcionamiento del sistema teleoperado diseñado en la sección anterior, los cuales incluyen: trayectorias en el espacio de trabajo, los efectos de aplicar fuerzas externas en la muñeca del brazo robótico, y manipulación de objetos; mostrando los resultados mediante fotografías, gráficos respecto al tiempo y trayectorias en el espacio cartesiano.

3.1. Trayectorias

El primer experimento, como es de suponer, abarca el desplazamiento por todo el espacio de trabajo de ambos dispositivos, tanto en posición como en orientación, así pues, en la Figura 3.1 se muestran fotografías desde distintos puntos de vista del sistema en ejecución, es justo aclarar que si bien el objetivo principal de este proyecto es realizar una teleoperación, se han situado ambos dispositivos juntos para mayor comprensión, pudiéndose así notar que el brazo robótico responde fielmente a los movimientos del maestro, además, es destacable la similitud de las imágenes con las mostradas en la Figura 2.14, correspondiente a la simulación mediante *MATLAB* y *VREP*.

Como se detalló en la sección dedicada a la interfaz, se ha dispuesto la opción para exportar los datos de la sesión en formato *.csv*, haciendo viable el análisis posterior de los mismos, por ejemplo, para describir la trayectoria realizada por el robot, como la expuesta en la Figura 3.2, que muestra el recorrido realizado por la muñeca del *MICO* en un espacio cartesiano en tres dimensiones, se alcanza a estimar que el error se encuentra en el orden de los milímetros, lo cual es suficiente para aplicaciones cotidianas, como es el objetivo de esta investigación.

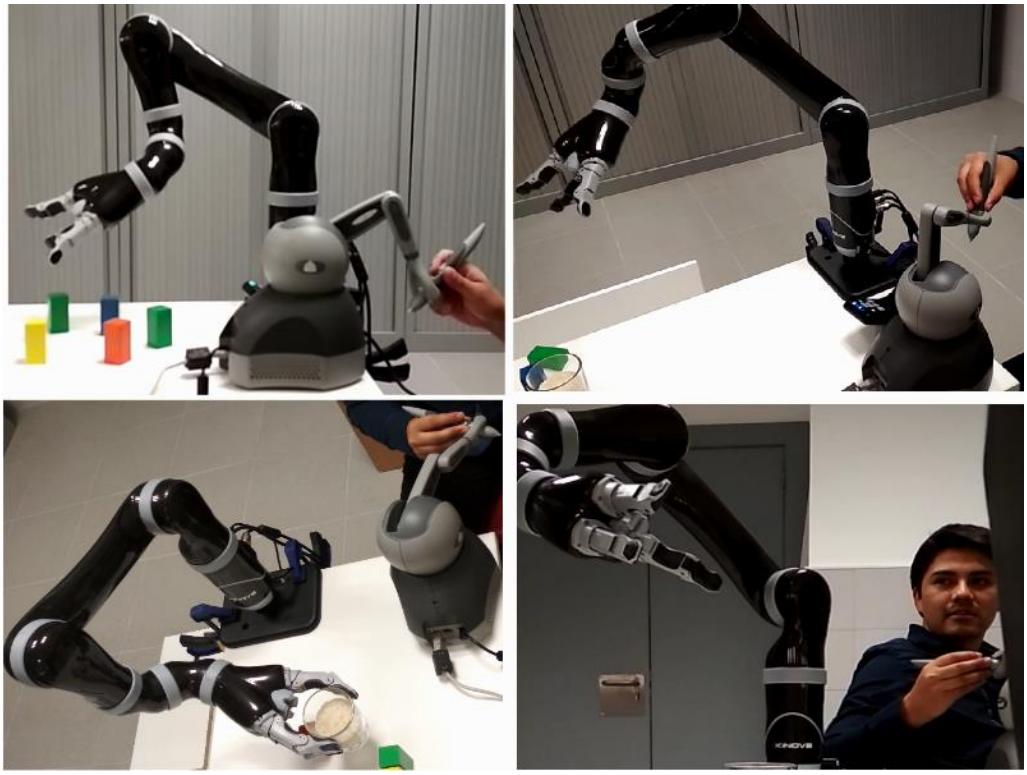


Figura 3.1. Fotografías del sistema en funcionamiento.

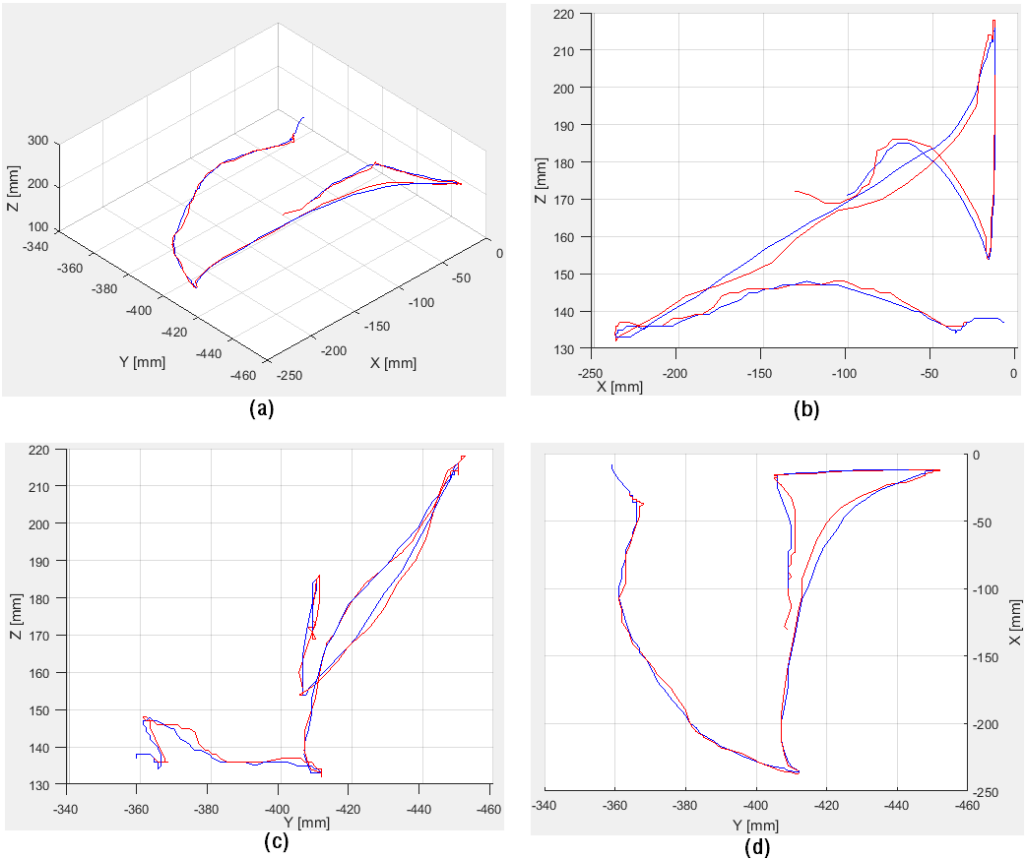


Figura 3.2. Trayectoria en espacio tridimensional, en rojo la posición deseada y en azul la real. (a) Vista isométrica, (b) Plano XZ, (c) Plano YZ y (d) Plano XY.

Otro método bastante usual de ilustrar los resultados, es graficar cada parámetro respecto al tiempo para ver su evolución, además de que se hace más apreciable el error que en un entorno tridimensional, tal y como se expone en la Figura 3.3, la cual muestra cada componente de traslación y orientación para la trayectoria mostrada en la Figura 3.2, adicionalmente se ha agregado el error promedio para cada una de ellas, calculada mediante la expresión siguiente.

$$Error = \sum_{k=1}^{n} \frac{|Posición_{Deseada}(k) - Posicion_{Real}(k)|}{n} \tag{10}$$

En cuanto a la traslación, se puede acotar que el desliz es menor al centímetro, incluso si se toma en cuenta que el cálculo del error está afectado por el desfase que existe entre ambas señales, ya que se puede valorar la similitud entre ambas si estuviesen en fase. Por otro lado, en orientación existe un error cercano a los diez grados sexagesimales, aquí es pertinente acotar que al brazo robótico le es más difícil adquirir una nueva orientación manteniendo la posición fija, por consiguiente, tarda más en llegar al valor consigna, y se ha preferido dejarlo con este comportamiento, ya que, al aumentar las ganancias de control para intentar reducirlo, se obtuvo el efecto contrario, llegando a la oscilación.

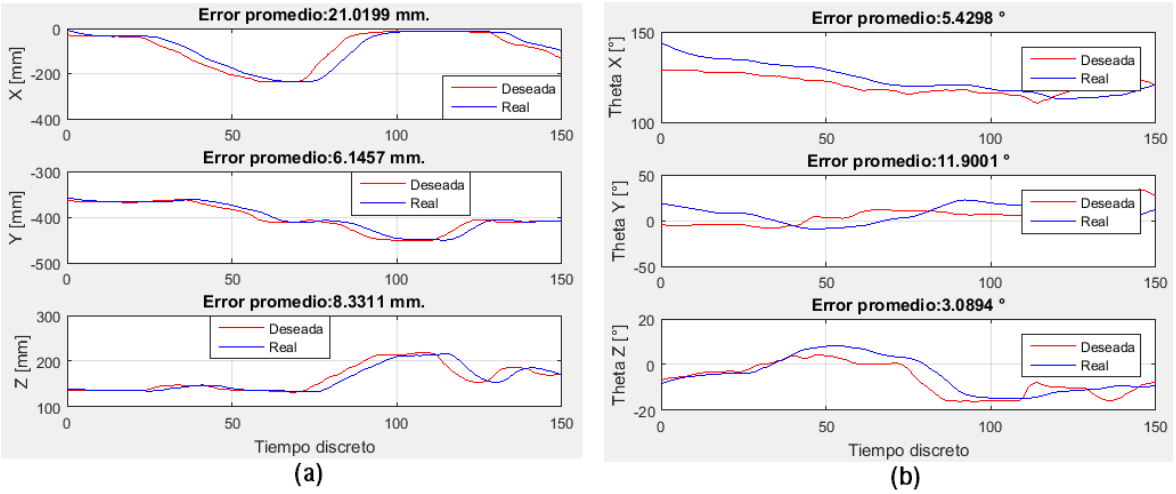


Figura 3.3. Trayectoria respecto al tiempo. (a) Traslación y (b) Orientación.

3.2. Realimentación de fuerzas

A pesar de que resulta difícil exponer los efectos de la perturbación de fuerzas en un documento, en la Figura 3.4 se intenta ilustrar el resultado que se tiene al empujar el efector final del brazo robótico sobre el dispositivo háptico, como es observable en los recuadros marcados de color rojo, existe un desplazamiento

debida a la fuerza realimentada que el brazo robótico está percibiendo, y al ser un control bilateral, la nueva posición afectará a la consigna que será enviada posteriormente al robot, lo cual será más apreciable en un gráfico en el espacio cartesiano.

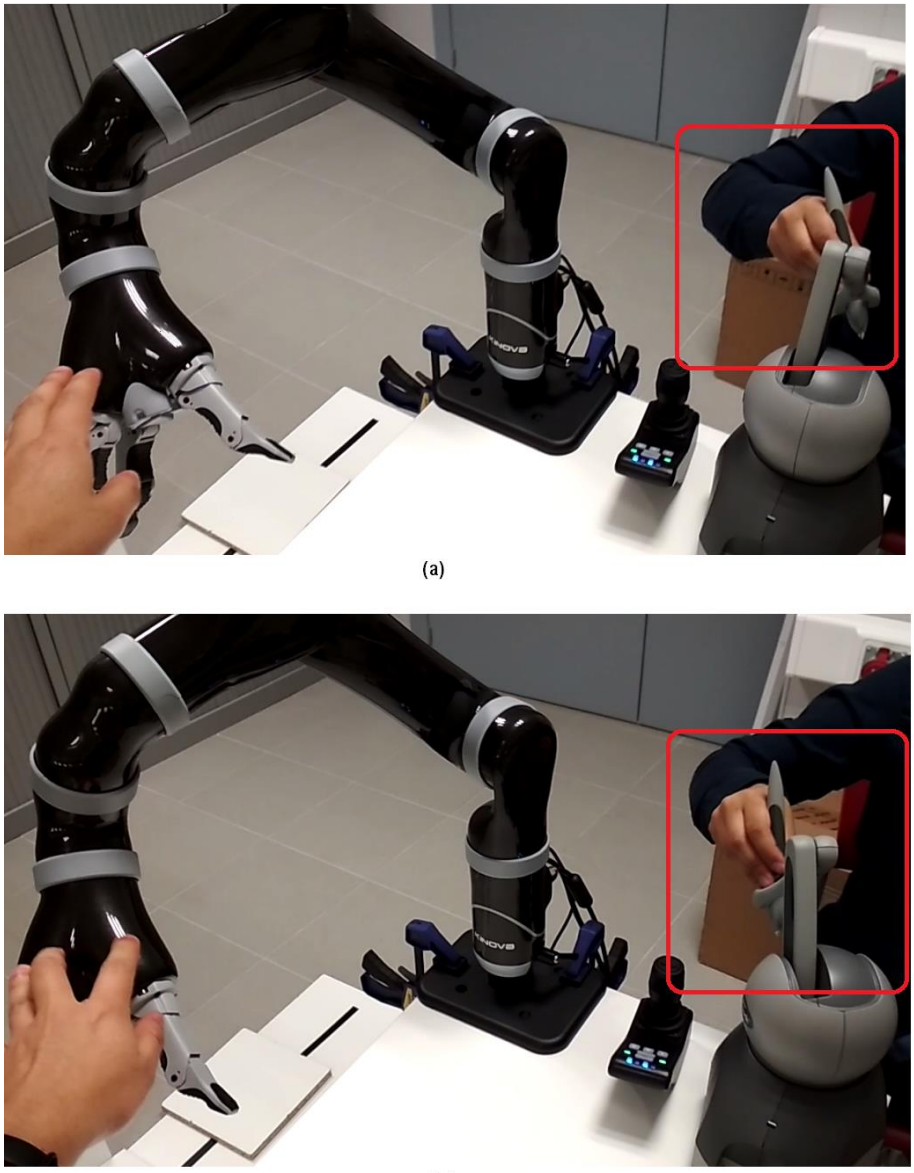


Figura 3.4. Fotografías de la respuesta a fuerzas externas. (a) Antes de aplicar fuerza al efector y (b) Después de la perturbación de fuerzas.

En la Figura 3.5 se observan los resultados antes comentados en el espacio tridimensional al aplicar una fuerza externa en la muñeca del brazo robótico, es claramente visible el desfase que ocurre en el espacio debido a la perturbación; teóricamente y dada la realimentación tanto de fuerza como de posición, no debería existir dicho desfase, pues idealmente ambos deberían reaccionar inmediatamente, pero ya que en el sistema real existen tiempos de retardos debido a diferentes causas

como el período de muestreo, comunicación, etc., el resultado es como el mostrado en las imágenes mencionadas.

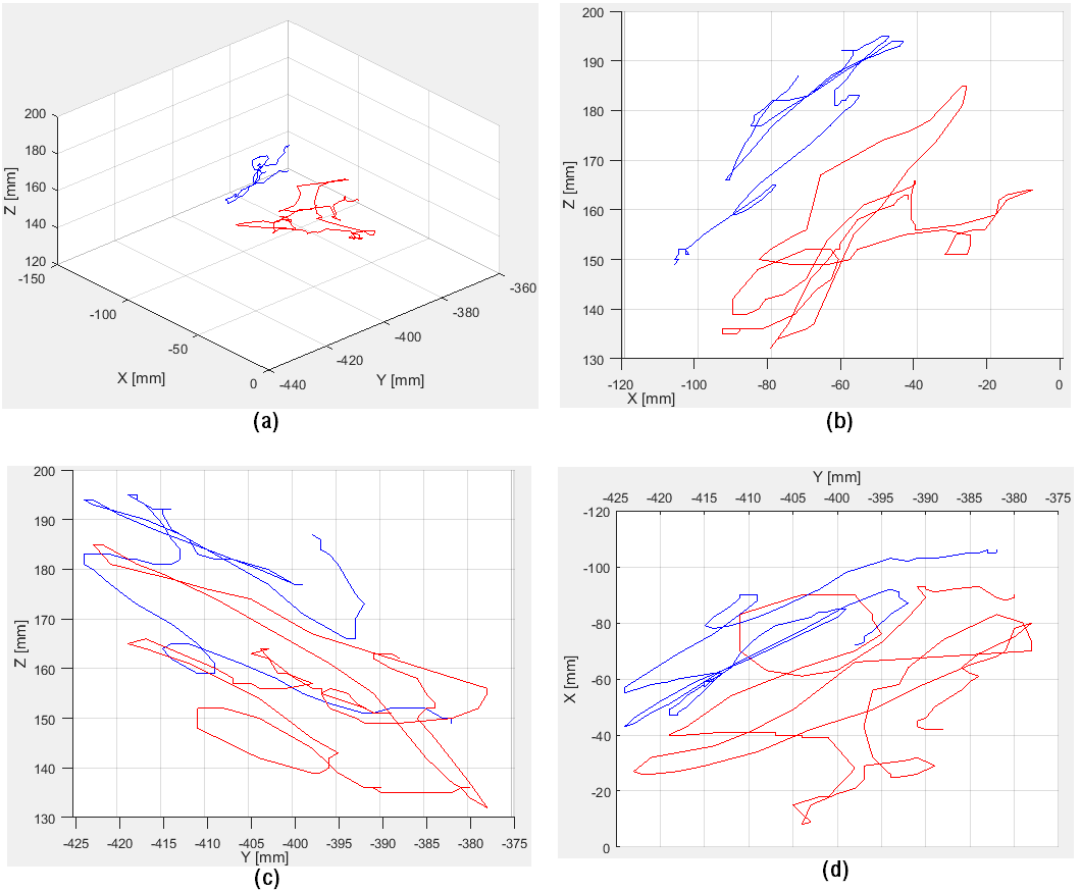


Figura 3.5. Efectos de la perturbación de fuerza mostrado en el espacio cartesiano. (a) Vista isométrica, (b) Plano XZ, (c) Plano XZ y (d) Plano XY.

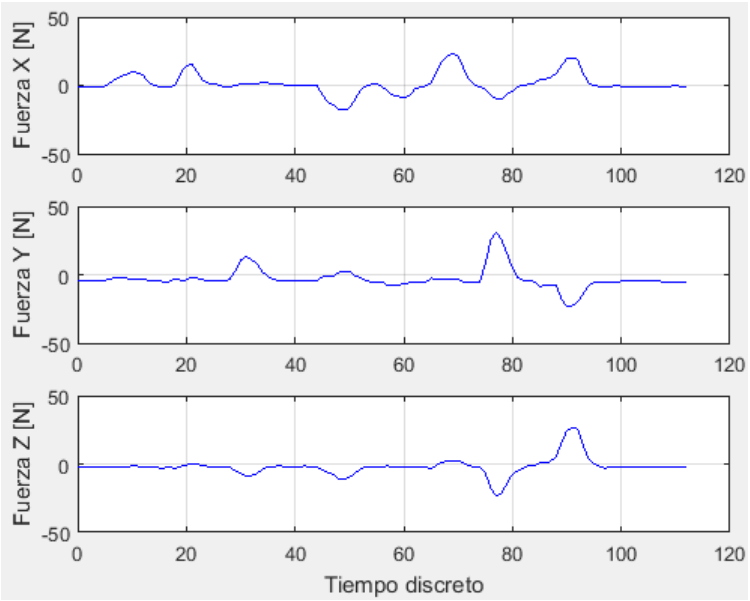


Figura 3.6. Componentes de fuerza medida en el efector del brazo robótico.

Las fuerzas de perturbación se exponen en la Figura 3.6, cabe recordar que el brazo *MICO* es capaz de soportar magnitudes de fuerza mayores a los que el dispositivo háptico puede generar, como en este caso particular, en el que la fuerza se encuentra alrededor de 25 N., por eje, y el *Phantom* solo es capaz de producir 3.3 N., motivo por el cual existe un factor de escalamiento, que para este experimento fue de 5, por tanto, se aplicó la máxima realimentación con estas perturbaciones.

3.3. Manipulación de objetos

Finalmente, se ha realizado pruebas mediante la manipulación de distintos objetos, entre ellos, piezas de madera, como las fotografías en la Figura 3.7 muestran. Es apropiado enfatizar que se obtuvieron buenos resultados, tanto en agarre, precisión y facilidad de manejo, pues resulta más intuitivo manipular un dispositivo maestro en tres dimensiones, que un panel de control o incluso un joystick.

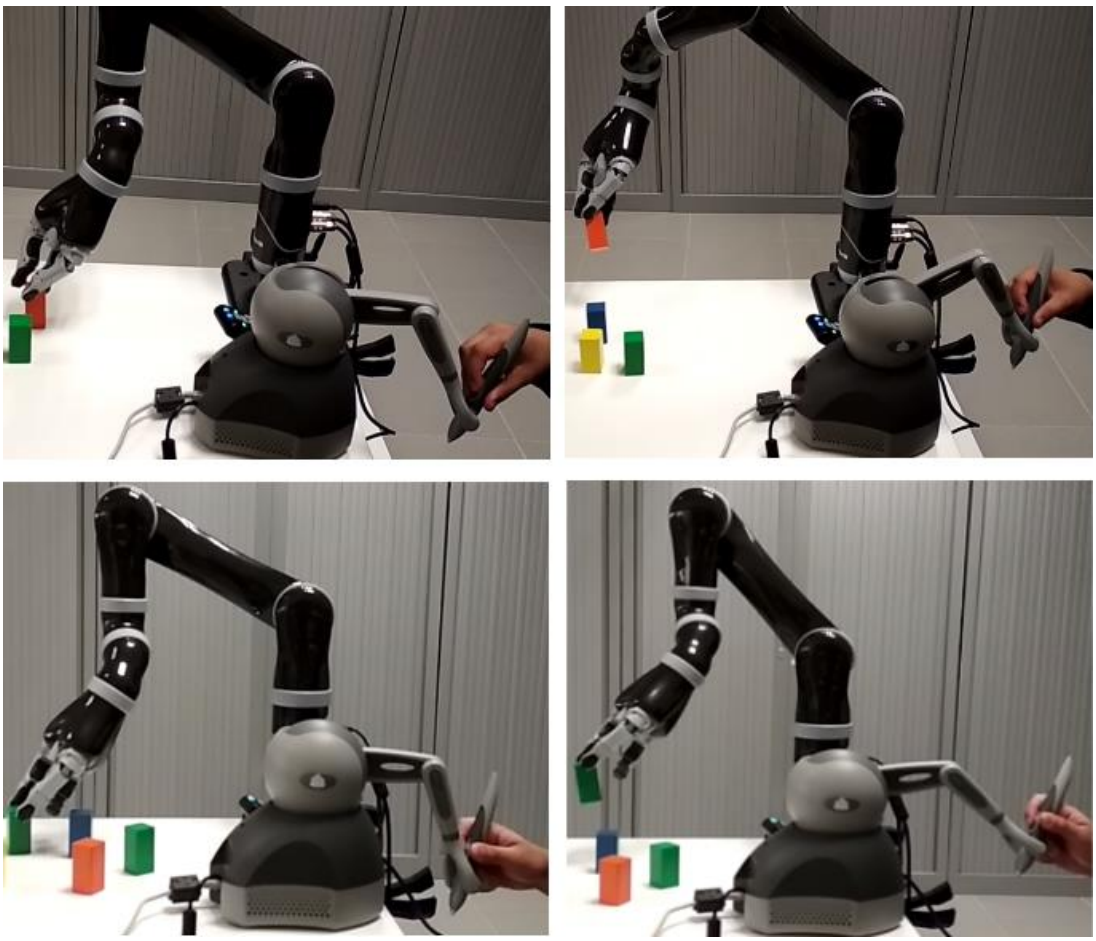


Figura 3.7. Manipulación de piezas de madera.

Con los resultados obtenidos de los experimentos anteriores, se decide manipular objetos más frágiles, como un vaso de vidrio por ejemplo, lo cual se puede observar en la Figura 3.8; se lo ha llenado con arroz por seguridad, ya que cualquier líquido puede resultar dañino para los dispositivos electrónicos, en caso de algún fallo. Igual que el caso antes descrito, los resultados son satisfactorios, mostrando un gran nivel de operatividad y cumpliendo apropiadamente los objetivos planteados para un uso en entornos cotidianos, propuestos en este proyecto.

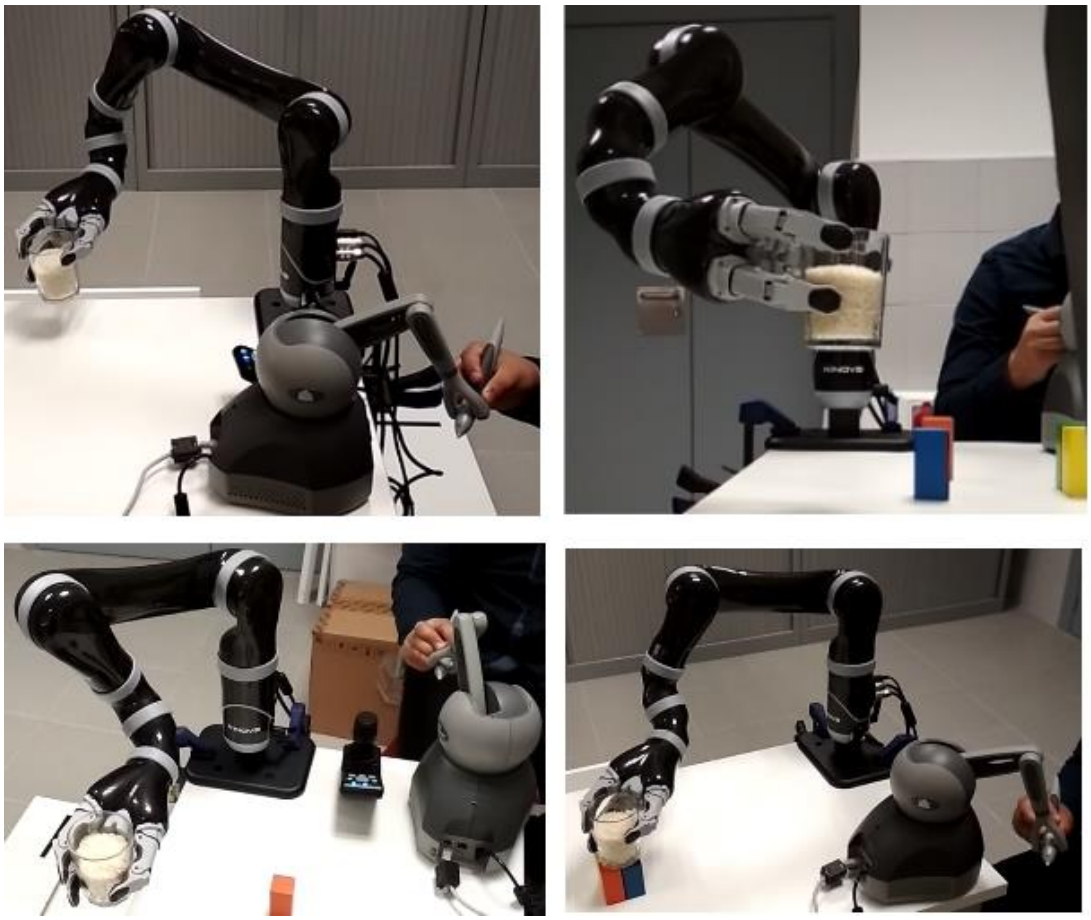


Figura 3.8. Manipulación de un vaso de vidrio.

Capítulo 4

Conclusiones

Por último, en este capítulo se presentarán las conclusiones a las que se han llegado al ejecutar esta investigación, conjuntamente, se propondrá trabajos que pudiesen derivar, fruto de los resultados obtenidos en el mismo.

4.1. Conclusiones

Los sistemas teleoperados surgieron como necesidad de manipular objetos peligrosos sin exponer a las personas a dichos riegos, a pesar de ello, la tecnología ha ido explorando nuevas aplicaciones para estos sistemas, y como es el caso de este estudio, en aplicaciones de uso común donde la robótica está ganando campo a diario. Con los resultados logrados, se presenta este método como válido para la teleoperación de un brazo robótico, dado que ha demostrado cumplir con especificaciones suficientes para ser manipulado y operado dentro de un régimen doméstico o comercial.

Mediante la experimentación y el uso de este sistema, se concluye que el manejo por parte de un dispositivo háptico maestro presenta mayor operatividad que otras interfaces de control como pueden ser: *joystick* o paneles de control ya sean físicos o virtuales; puesto que el manejo de una herramienta en el espacio tridimensional resulta más intuitivo para un ser humano, a diferencia de objetos estáticos como botones, controles deslizantes o palancas, que no replican de manera exacta los movimientos que el efector del robot describirá. Así como también, por tener una referencia de la fuerza soportada por el manipulador, que permite percibir golpes, choques con objetos, o incluso el peso del objeto a operar.

Uno de los pilares de un sistema teleoperado es la comunicación, en esta investigación se ha validado el uso del protocolo UDP como apto para el intercambio de información entre dispositivos maestro y esclavo, lo que amplía el rango de teleoperación a una escala global, sin olvidar las peculiaridades que esto conlleva como tiempos de retardo dictados por el funcionamiento de la red.

4.2. Trabajos futuros

El brazo robótico *MICO*, como se mencionó, permite además un control articular, tanto de posición como de velocidad y torque; lo cual podría ser otro método de mando para verificar su rendimiento respecto al propuesto en este documento, esto conlleva por supuesto a que se deba realizar la cinemática inversa, pero dado que se tiene un control a menor nivel, podría ofrecer mejores resultados o al menos, se tendría mayor dominio sobre el brazo robótico.

Dentro de los dispositivos hápticos, como se expuso en el capítulo introductorio, existen varios tipos, de tal modo que se podría utilizar un guante háptico como dispositivo maestro, para estudiar las diferencias entre ambos dispositivos, así como ventajas o inconvenientes de cada uno, y aplicaciones en las cuales serían aplicables.

Apéndice A

Instalación y configuración de SDK para Visual Studio Community 2017

Esta guía pretende ser una ayuda en la configuración de las herramientas de desarrollador ofrecidas por los proveedores de los dispositivos usados en este proyecto, y no un tutorial desde cero de las mismas, ya que los fabricantes ofrecen documentación al respecto en sus páginas web [22], [23]. Por tanto, se asume que los controladores están instalados y funcionando correctamente sobre el sistema operativo, así como el software Visual Studio Community 2017 [24].

Cabe mencionar que las aplicaciones realizadas para este proyecto se generaron para arquitecturas de 32 bits, dando así mayor compatibilidad al software desarrollado.

A.1. Instalación de OpenHaptics

La versión utilizada en este trabajo es OpenHaptics® Toolkit Developer Edition v3.4, que puede ser descargada de la página web de 3D Systems [23], y se desarrolló el código sobre Visual Studio Community 2017. El procedimiento está basado en una guía de instalación para la versión de Visual Studio 2010 [25], adaptado para la versión 2017 sobre Windows® 7 de 64 bits.

Como primera consideración, el SDK está fundamentado en Visual Studio 2010, por lo cual es necesario tener los compiladores de esta versión instalados, que se pueden obtener en la página de Microsoft [26]. Los paquetes a instalar son: Visual Studio 2010 VC Express y Visual Studio 2010 Service Pack 1.

Se realiza la instalación del software en la ruta *C:\OpenHaptics\Developer\3.4.0* como indica la Figura A.1, es importante destacar que se usará está ruta en el resto de la configuración, por tanto, si se ha elegido otra dirección de instalación tenerlo en cuenta para los siguientes pasos.

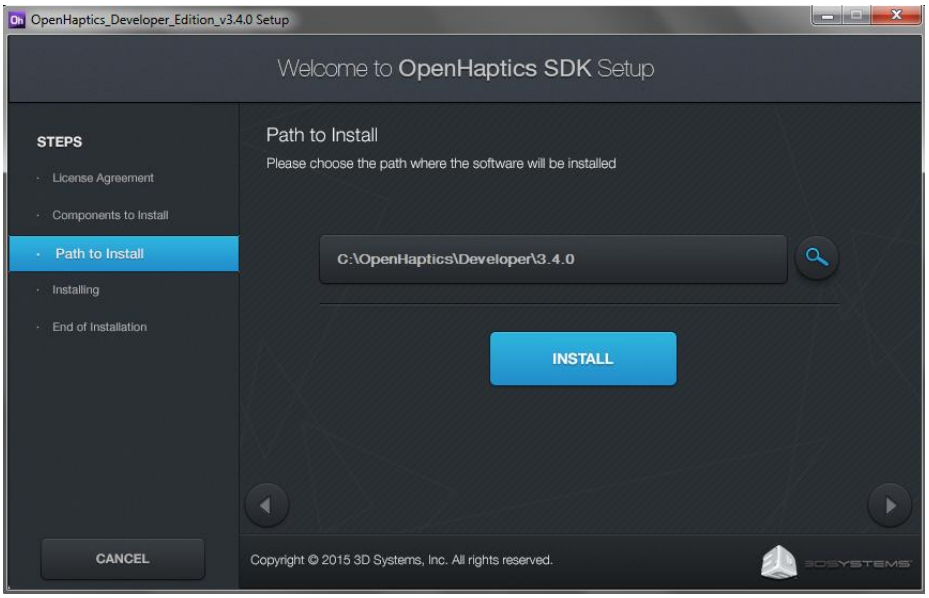


Figura A.1. Ruta de instalación de OpenHaptics SDK.

Abrimos nuestro proyecto de Visual Studio, y en la pestaña *Project*, seleccionamos la opción P*roperties*. Como se muestra en la Figura A.2.

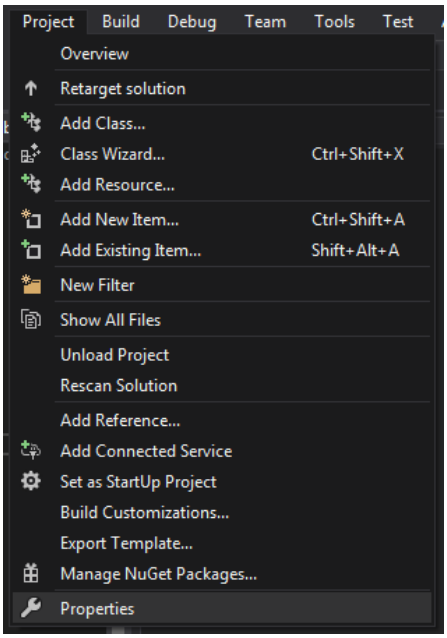


Figura A.2. Propiedades del proyecto en Visual Studio.

En la ventana que se abre del paso anterior, ir a la configuración *General -> Project Defaults -> Character Set* como ilustra la Figura A.3. Para tener todas las funciones ofrecidas por OpenHaptics, como QuickHaptics o interfaces en Win32 o GLUT seleccionar la opción *Use Multi-Byte Character Set*, pero esta configuración no es compatible con el SDK de Kinova, por tanto, si se va a utilizar el Phantom y el MICO en el mismo proyecto, seleccionar *Use Unicode Character Set*. Además, seleccionar la opción *<inherit from parent or project defaults>* de la lista de *Platform Toolset*.

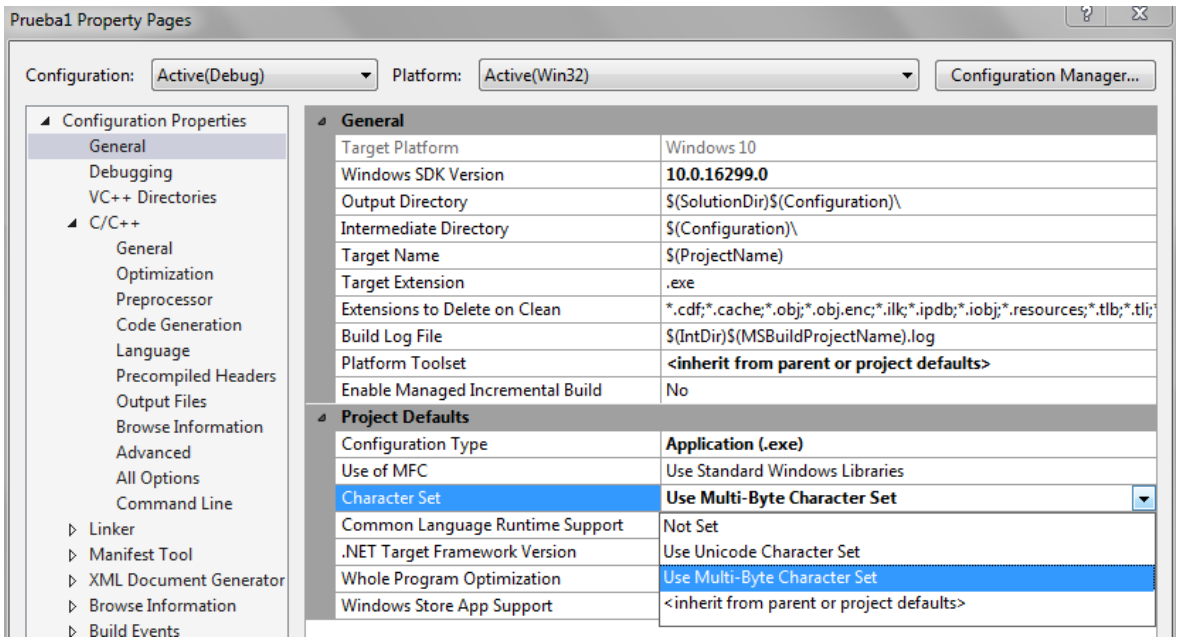


Figura A.3. Configuración de *Character Set*.

Ahora elegir la opción *VC++ Directories*, donde se va a incluir las siguientes líneas en los apartados *Include Directories* y *Library Directories* respectivamente. En la Figura A.4 se observa el resultado.

C:\OpenHaptics\Developer\3.4.0\include\HL;C:\OpenHaptics\Developer\3.4.0\include\HD;$(IncludePath)
C:\OpenHaptics\Developer\3.4.0\Quickhaptics\lib\Win32\Release;C:\OpenHaptics\Developer\3.4.0\lib\Win32\Release;$(LibraryPath)

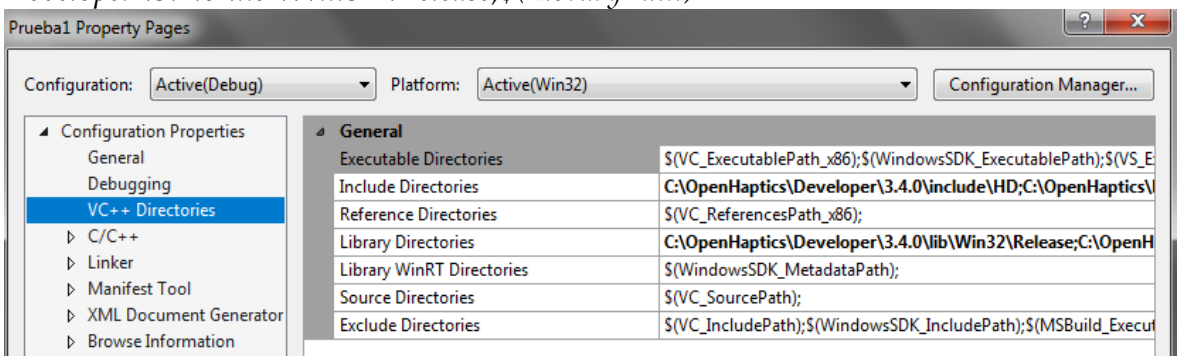


Figura A.4. Configuración de *VC++ Directories*.

Desplegar la pestaña *C/C++* y optar por *General*, insertar la siguiente línea en la opción *Additional Include Directories*, como muestra la Figura A.5.

include;$(OH_SDK_BASE)\include;$(OH_SDK_BASE)\utilities\include;$(OH_SDK_BASE)\QuickHaptics\header;%(AdditionalIncludeDirectories)

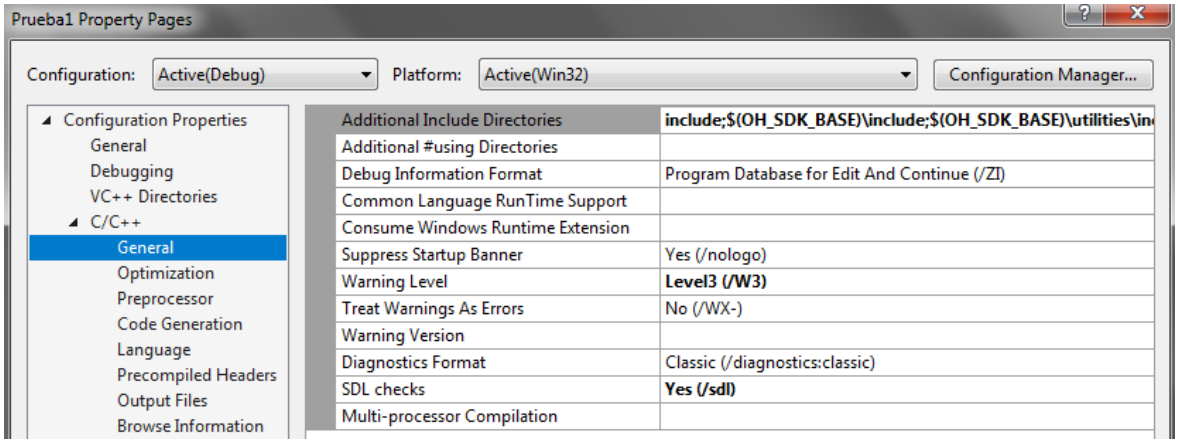


Figura A.5. Configuración de *Additional Include Directories*.

Dentro de la misma pestaña anterior, ahora seleccionar *Code Generation* y en la lista desplegable de *Runtime Library*, escoger la opción *Multi-threaded DLL (/MD)* o *Multi-threaded Debug DLL (/MDd)* de acuerdo a la configuración del proyecto, ya sea Release o Debug, respectivamente.

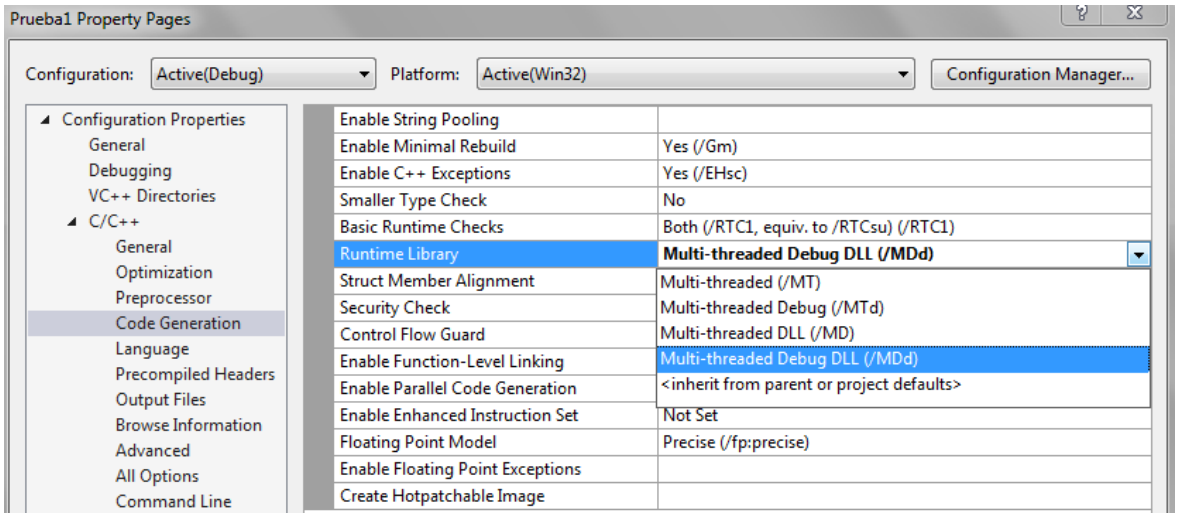


Figura A.6. Configuración de *Runtime Library*.

Expandir el menú *Linker*, seleccionar *General* y agregar lo indicado a continuación en *Additional Library Directories*. Tal y como se ve en la Figura A.7.

$(OH_SDK_BASE)\lib\$(Platform)\$(Configuration);$(OH_SDK_BASE)\utilities\lib\$(Platform)\$(Configuration);$(OH_SDK_BASE)\QuickHaptics\lib\$(Platform)\$(Configuration);%(AdditionalLibraryDirectories)

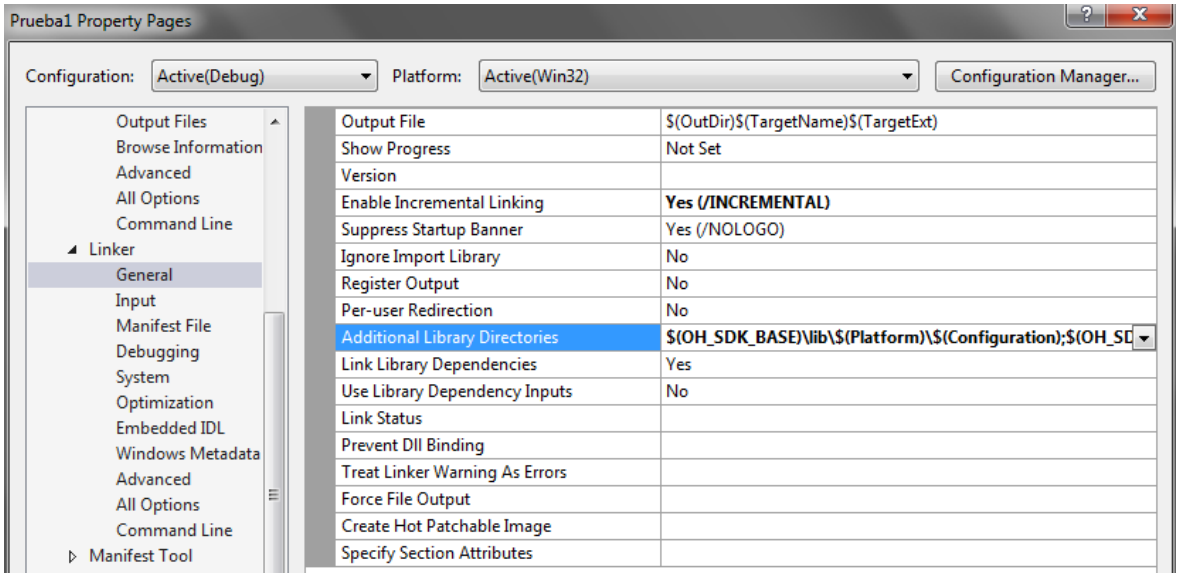


Figura A.7. Configuración de *Linker (General)*.

En el mismo menú anterior, elegir *Input* y agregar lo indicado a continuación en *Additional Dependencies*. El resultado se observa en la Figura A.8.

hd.lib;hdu.lib;hl.lib;hlu.lib;hapticmouse.lib;snapconstraints.lib;QH.lib;QHWin32Wrapper.lib;QHGLUTWrapper.lib;glui32.lib;glut32.lib

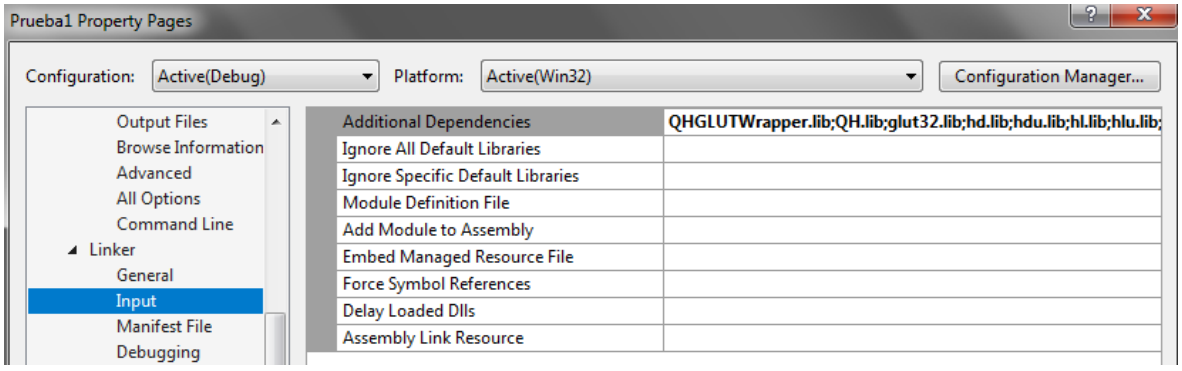


Figura A.8. Configuración de *Linker (Input)*.

La configuración está completa. A continuación, se detalla soluciones a problemas que se puede tener en la compilación o ejecución del código.

En el caso de utilizar la interfaz gráfica que ofrece el SDK, se puede optar por GLUT o WIN32, para utilizar el primero se debe crear el proyecto como *Console application*, o *Windows application* si se quiere la segunda opción. De manera que si se obtiene el error:

unresolved external symbol winmain@16 referenced in function ___

Es porque no se eligió adecuadamente el tipo de aplicación, se lo puede cambiar en *Linker -> System -> Subsystem*. En la Figura A.9, se presenta la ventana.

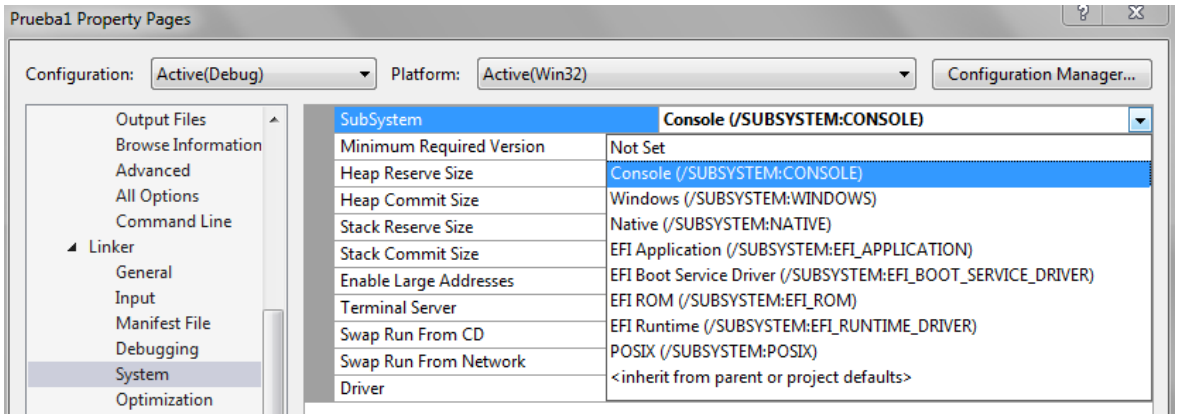


Figura A.9. Elección de tipo de aplicación.

En caso de que el software indique la falta del archivo *msvcp120d.dll*, se puede copiar el mismo en la carpeta del proyecto, o copiarlo en la dirección del sistema operativo, y así quedará disponible para todos los proyectos. El *path* es el siguiente:

C:\Windows\SysWOW64

A.2. Instalación de Kinova SDK

La versión empleada en la presente investigación es la MICO2 1.4.0, que se obtuvo de la página oficial de Kinova [27].

Al igual que en el apartado anterior, se utilizará la dirección de instalación mostrada en la Figura A.10 para configurar el SKD del robot MICO. Por ende, si se elige otra ruta, tomarlo en cuenta en los siguientes pasos.

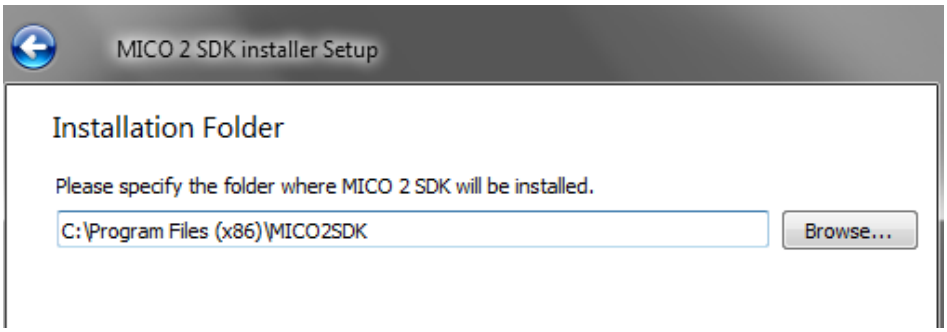


Figura A.10. Ruta de instalación para MICO 2 SDK.

Una vez instalado, en la ubicación detallada en la Figura A.11 encontraremos los archivos *.dll* y las librerías *.h* necesarios para cada arquitectura, dentro de su respectiva carpeta. La arquitectura dependerá de la configuración de la aplicación a realizar en Visual Studio, no necesariamente siendo la misma que del sistema operativo.

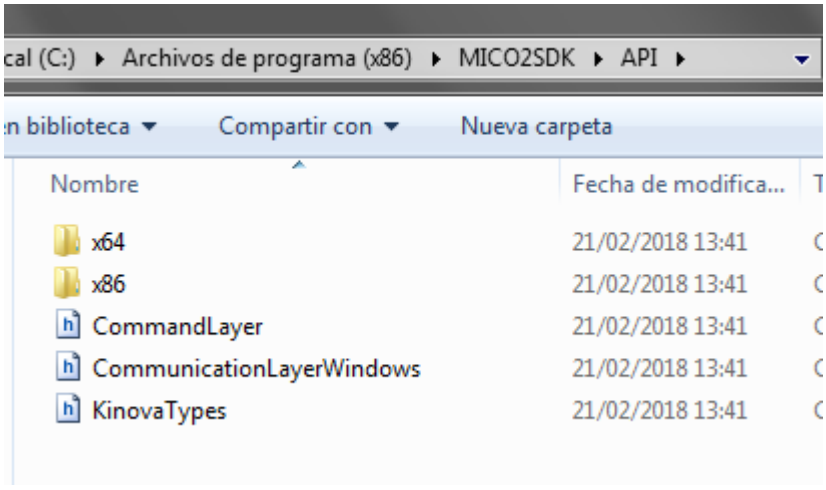


Figura A.11. Archivos *.dll* de MICO 2 SDK.

Existen dos opciones para configurar el proyecto, la primera es agregar los archivos antes mencionados en el proyecto directamente, como indica la Figura A.12. Y eso sería todo.

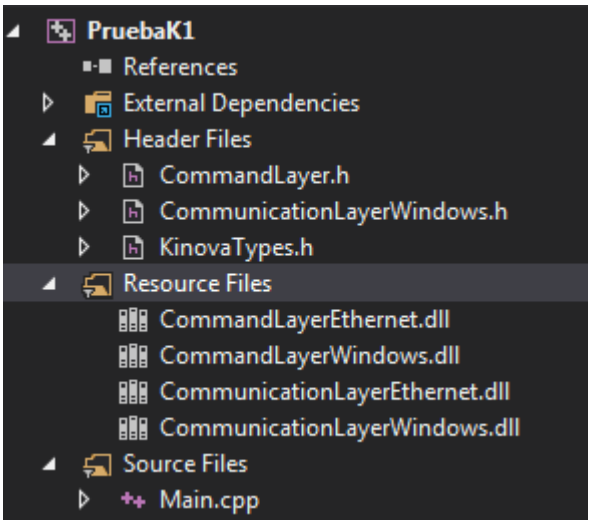


Figura A.12. Añadir librerías y recursos al proyecto.

La segunda opción es pegar los archivos *.dll* en las siguientes direcciones:

Para arquitectura de 64 bits: *C:\Windows\System32*

Para arquitectura de 32 bits: *C:\Windows\SysWOW64*

Y agregar el siguiente *path* en la configuración del proyecto. En *VC++ Directories*, en la opción *Include Directories.* Como la Figura A.13 ejemplifica.

C:\Program Files %28x86%29\MICO2SDK\API;$(IncludePath)

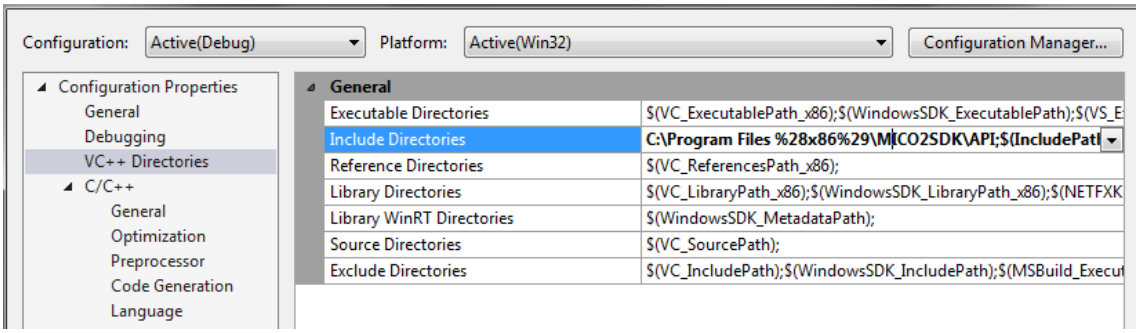


Figura A.13. Configuración de *VC++* para MICO 2 SDK.

Bibliografía

[1] E. Nuño Ortega and L. Basañez, "Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente," pp. 1–43, 2004.

[2] M. Mihelj and J. Podobnik, *Haptics for Virtual Reality and Teleoperation*, vol. 64. Springer, 2012.

[3] D. Pfeil-Seifer and Mataric, "Defining socially assistive robotics," *Proceedings*, pp. 465–468, 2005.

[4] H. L. Martin and W. R. Hamel, "Joining teleoperation with robotics for advanced manipulation in hostile environments," 1984.

[5] D. A. Lawrence, "Stability and Transparency in Bilateral Teleoperation," *IEEE Trans. Robot. Autom.*, vol. 9, no. 5, pp. 624–637, 1993.

[6] J. H. Ryu, J. Artigas, and C. Preusche, "A passive bilateral control scheme for a teleoperator with time-varying communication delay," *Mechatronics*, vol. 20, no. 7, pp. 812–823, 2010.

[7] R. N. Jazar, *Theory of Applied Robotics*. Springer, Boston, MA, 2010.

[8] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Auton. Robots*, vol. 11, no. 1, pp. 9–18, 2001.

[9] T. Kot and P. Novák, "Application of virtual reality in teleoperation of the military mobile robotic system TAROS," *Int. J. Adv. Robot. Syst.*, vol. 15, no. 1, 2018.

[10] D. Ni, A. W. W. Yew, S. K. Ong, and A. Y. C. Nee, "Haptic and visual augmented reality interface for programming welding robots," *Adv. Manuf.*, vol. 5, no. 3, pp. 191–198, 2017.

[11] E. Olivieri, G. Barresi, D. G. Caldwell, and L. S. Mattos, "Haptic Feedback for Control and Active Constraints in Contactless Laser Surgery : Concept , Implementation and Evaluation," vol. 1412, no. c, pp. 1–14, 2017.

[12] 3D Systems, "The Touch™ Haptic Device," 2018. [Online]. Available: https://es.3dsystems.com/haptics-devices/touch. [Accessed: 01-May-2018].

[13] A. Nygaard, "High-Level Control System for Remote Controlled Surgical Robots," 2008.

[14] M. H. Koul, P. Kumar, P. K. Singh, M. Manivannan, and S. K. Saha, "Gravity Compensation for PHANToM TM Omni ® Haptic Interface," no. May, 2010.

[15] A. Jarillo-Silva, O. A. Domínguez-Ramírez, V. Parra-Vega, and J. P. Ordaz-Oliver, "PHANToM OMNI haptic device: Kinematic and manipulability," *CERMA 2009 - Electron. Robot. Automot. Mech. Conf.*, pp. 193–198, 2009.

[16] C. Fulford *et al.*, "Quanser Engineering Blog," 2012. [Online]. Available: http://quanser.blogspot.com.es/2012/05/. [Accessed: 02-May-2018].

[17] Kinova, "Robot arms." [Online]. Available: http://www.kinovarobotics.com/innovation-robotics/products/robot-arms/. [Accessed: 02-May-2018].

[18] Quanser, "Kinova 6-DOF MICO Read." [Online]. Available: http://quanser-update.azurewebsites.net/quarc/documentation/kinova_6dof_mico_read_block.html. [Accessed: 02-May-2018].

[19] J. J. Scala Estalella, *Análisis vectorial*. Barcelona: Reverté, 1988.

[20] E. G. Gutiérrez, "Introducción al filtrado digital," no. June, pp. 10–12, 2010.

[21] E. Soria Olivas, M. Martínez Sober, J. V. Francés Villora, and G. Camps Valls, *Tratamiento Digital de señales.*, vol. 53, no. 9. 2003.

[22] Kinova, "SDK Development center User guide," 2017. [Online]. Available: http://www.kinovarobotics.com/wp-content/uploads/2017/10/Kinova-SDK-Development-Center-User-Guide.pdf. [Accessed: 20-Apr-2018].

[23] 3D Systems, "3D Systems Inc. - OpenHaptics for Windows Developer Edition v3.4." [Online]. Available: https://3dsystems.teamplatform.com/pages/102774?t=r4nk8zvqwa91. [Accessed: 16-Apr-2018].

[24] Microsoft, "Visual Studio." [Online]. Available: https://www.visualstudio.com/es/downloads/. [Accessed: 20-Apr-2018].

[25] M. Rayl, "OpenHaptics for Visual C++ 2010," 2010. [Online]. Available: http://infohost.nmt.edu/~mecheng/ril/includes/resources/OpenHapticsHelp.pdf. [Accessed: 30-Jan-2018].

[26] Microsoft, "Visual Studio VC 2010." [Online]. Available: https://my.visualstudio.com/Downloads?q=visual studio 2010&wt.mc_id=o~msft~vscom~older-downloads. [Accessed: 20-Apr-2018].

[27] Kinova, "KINOVA SDK MICO2_1.4.0.zip." [Online]. Available: https://drive.google.com/file/d/0B790iVm0vRTlREtNSXd6US1ZM2M/view. [Accessed: 20-Apr-2018].